UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Desarrollo de un telémetro por infrarrojos y un sistema de control sensorial para la detección de obstáculos de un robot móvil.

PROYECTO FIN DE CARRERA

INGENIERÍA TÉCNICA INDUSTRIAL: ELECTRÓNICA INDUSTRIAL.

AUTOR: Juan Antonio Vigara González.

TUTOR: Javier Fernández de Gorostiza Luengo.

OCTUBRE, 2009

RESUMEN

El presente proyecto se integra dentro del desarrollo del robot *Maggie* que se está llevando a cabo en el laboratorio de robótica de la Universidad Carlos III de Madrid. El proyecto se ha realizado como base para poder desarrollar en el robot la habilidad de evitar obstáculos, utilizando para ello los sensores que tiene integrados en su base, infrarrojos y ultrasonidos. Se ha diseñado un sistema que permite detectar obstáculos a diferentes distancias, desde pocos centímetros hasta varios metros, y determinar la posición del robot permitiéndole el movimiento por su entorno con autonomía.

ABSTRACT

The present project is integrated within the development of the robot *Maggie* being carried out in the laboratory of robotics of Carlos III University in Madrid. The project has been done as a basis in order to develop in the robot the ability to avoid obstacles, using the sensors is integrated into their base, infrared and ultrasound. It has been designed a system that allows to detect obstacles at different distances, from a few inches up to several meters, and determine the position of the robot allowing the movement by its environment with autonomy.

ÍNDICE

RESUMEN 2

ABSTRACT 3

CAPÍTULO 1.- INTRODUCCIÓN 14

1.1 Presentación y motivación 15

1.2 Objetivos 15

1.3 Los robots personales 16

1.4 *Maggie*, robot donde se integra el proyecto 19

1.4.1 Base del robot 20

1.4.2 Cuerpo del robot 21

1.4.3 Sistema sensorial 21

1.5 Evolución del mercado de robots sociales 22

CAPÍTULO 2.- ARQUITECTURA HARDWARE 24

2.1 Descripción y esquema general 25

2.2 Descripción hardware del telémetro por infrarrojos 27

2.2.1 Sensor Sharp GP2D02 27

2.2.1.1 Principio de funcionamiento del sensor: 29

2.2.1.2 Protocolo para la obtención de medidas: 31

2.2.1.3 Calibración del sensor: 33

2.2.2 Microcontrolador de los sensores IR: PIC16F876 35

2.2.2.1 Oscilador: 36

2.2.2.2 Circuito de Reset: 38

2.2.2.3 Memoria de datos SRAM: 39

2.2.2.4 Periféricos integrados en el micro y utilizados en este proyecto: 39

2.2.2.4.1 Puertos I/O: 39

2.2.2.4.2 Timer 0: 41

2.2.2.4.3 El puerto serie USART: 43

2.2.2.4.4 Módulo Master Synchronous Serial Port (MSSP): 46

2.2.2.5 Esquema de conexión de los terminales de los sensores a los puertos E/S del micro: 48

2.2.3 Conectores 49

2.2.3.1 Conexión a los sensores: 49

2.2.3.2 Conexión al PC o a la placa de control: 50

2.2.3.3 Alimentación: 50

2.2.4 Selección de la denominación de puerta 51

2.3 Integración del telémetro por ultrasonidos 52

2.4 Integración de una brújula digital (“módulo CMPS03”) 54

2.4.1 Principio de funcionamiento 54

2.5 Descripción hardware de la placa de control sensorial 57

2.5.1 Microcontrolador para la integración de los telémetros y la brújula: PIC18F2550 57

2.5.1.1 Memoria de datos SRAM: 59

2.5.1.2 Periféricos integrados en el micro y utilizados en este proyecto: 59

2.5.1.2.1 Puertos I/O: 59

2.5.1.2.2 El puerto serie EUSART: 60

2.5.1.2.3 Módulo Master Synchronous Serial Port (MSSP): 60

2.5.2 Conectores 62

2.5.2.1 Conexión de la placa de control con telémetros sonar e IR: 62

2.5.2.2 Conexión de la placa de control con la brújula: 62

2.5.2.3 Conexión de la placa de control con el PC del robot: 63

2.5.2.4 Alimentación: 64

2.5.3 Sistema de visualización del funcionamiento del sistema 64

2.6 Protocolo de comunicación entre dispositivos. Nivel físico 66

2.6.1 Estándar de comunicación I^2C 66

2.6.2 Estándar de comunicación RS-232 67

2.6.2.1 Adaptador de tensiones: 68

2.6.2.2 Adaptador RJ45-DB9: 68

2.6.2.3 Adaptador Serie DB9 - USB 2.0: 69

CAPÍTULO 3.- IMPLEMENTACIÓN DEL SOFTWARE 70

3.1 Protocolos de comunicación. Tramas de datos 71

3.1.1 I^2C 71

3.1.1.1 Maestro envía datos al esclavo: 72

3.1.1.2 Maestro recibe datos del esclavo: 73

3.1.1.3 Maestro envía/recibe datos del esclavo: 73

3.1.2 RS-232 74

3.2 Programas 74

3.2.1 Telémetro por infrarrojos 75

3.2.1.1 Recepción/envío de datos de/a la placa de control: 76

3.2.1.2 Realizar operación: 78

3.2.1.3 Realizar medida: 81

3.2.2 Placa de control 82

3.2.2.1 Recibir posición de la brújula: 86

3.2.2.2 Envío/recepción de datos a/de los dispositivos esclavos: 87

3.2.2.3 Envío de resultados al PC. 88

3.2.2.3.1 Informe de errores. 88

3.3 Implementación del driver (controlador). 91

3.3.1 Abrir y configurar puerto. 91

3.3.2 Enviar cadena. 92

3.3.3 Recibir cadena. 93

3.3.4 Obtener orientación y distancia al objeto. 94

3.3.4.1 Distancia a los sensores infrarrojos: 94

3.3.4.2 Distancia a los sensores ultrasonidos: 94

3.3.4.3 Orientación de la brújula: 94

3.3.5 Cálculo código de comprobación de datos sensores infrarrojos y sonar.. 95

3.3.6 Cerrar puerto. 96

3.3.7 Programa de prueba. 97

CAPÍTULO 4.- RESULTADOS EXPERIMENTALES. 99

4.1 Telémetro por infrarrojos. 100

4.1.1 Calibración del sensor. 100

4.1.2 Efecto de la reflectividad en la toma de medidas. 102

4.1.3 Cálculo del valor de las constantes Kg y Ko. 105

4.1.4 Realización de medidas. 106

4.1.4.1 Errores en la medida: 108

4.2 Brújula digital. 109

4.2.1 Calibración de la brújula. 109

4.2.2 Realización de medidas. 110

4.3 Sistema general. 111

4.3.1 Frecuencia en la obtención de medidas. 114

4.3.2 Restricciones del sistema general. 115

CAPÍTULO 5.- CONCLUSIONES Y TRABAJOS FUTUROS 117

5.1 Conclusiones. 118

5.2 Trabajos futuros. 118

5.2.1 Integración de todo el sistema en una sola placa. 118

5.2.2 Comunicación al PC del robot mediante USB. 119

5.2.3 Comunicación del sistema sin cables (wireless) 119

5.2.4 Sustituir los sensores de infrarrojos. 119

ANEXOS 120

ANEXO A: DATASHEETS. 121

SENSOR GP2D02 122

DIODO 1N4148 126

PIC 16F876 130

PIC 18F2550 134

COMPASS MODULE CMPS03 139

ANEXO B: ESQUEMÁTICO Y LAYOUT 143

TELÉMETRO POR INFRARROJOS 144

PLACA DE CONTROL 147

ANEXO C: CÓDIGO 150

TELÉMETRO POR INFRARROJOS PUERTA DELANTERA 151

TELÉMETRO POR INFRARROJOS PUERTA TRASERA 162

PLACA DE CONTROL 173

ANEXO D: FUNCIONES Y DIRECTIVAS DEL COMPILADOR CCS 188

ANEXO E: GUÍA DE FUNCIONAMIENTO 192

BIBLIOGRAFÍA 194

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Robots personales. 17

Ilustración 2: Robot móvil personal *Maggie*. 19

Ilustración 3. Puertas frontal y trasera de la base del robot. 20

Ilustración 4: Distribución de los sensores en la base de Maggie. 22

Ilustración 5. Esquema general del sistema. 25

Ilustración 6: Medidor sensores infrarrojos. 27

Ilustración 7: Sharp GP2D02. 28

Ilustración 8: Terminales del Sharp GP2D02. 28

Ilustración 9: Conexión PIC16F876 a GP2D02. 29

Ilustración 10: Funcionamiento Sharp GP2D02. 30

Ilustración 11: Espectro electromagnético. 31

Ilustración 12: Vin y Vout para conseguir una medida. 32

Ilustración 13: Relación entre la distancia dada por el sensor (DEC) y la distancia real (L). 33

Ilustración 14: PIC16F876 y circuito oscilador. 35

Ilustración 15: Circuito de reset del PIC16F876. 38

Ilustración 16: Diagrama de bloques de transmisión de datos (USART). 45

Ilustración 17: Diagrama de bloques de recepción de datos (USART). 46

Ilustración 18: Diagrama de bloques módulo I^2C esclavo. 47

Ilustración 19: Conexiones sensores a los puertos del micro. 48

Ilustración 20: Diodo 1N4148. 49

Ilustración 21: Conectores a sensores. 49

Ilustración 22: Conector y cable plano de 16 hilos. 50

Ilustración 23: Conector medidor-placa de control o medidor-PC..... 50

Ilustración 24: Puertas A o B de la base del robot..... 51

Ilustración 25: Telémetro por ultrasonidos..... 52

Ilustración 26: Orientación de la brújula digital (Módulo CMPS03). 55

Ilustración 27: Patillaje del módulo CMPS03. 55

Ilustración 28: Placa de control. 57

Ilustración 29: PIC18F2550 y circuito oscilador..... 58

Ilustración 30: Diagrama de bloques módulo I^2C maestro..... 61

Ilustración 31: Conectores de la placa de control con telémetros. 62

Ilustración 32: Conector placa control-brújula. 63

Ilustración 33: Terminales conector DB9..... 63

Ilustración 34: Conexión de los LEDs al PIC18F2550. 64

Ilustración 35: Selección del tipo de comunicación. 66

Ilustración 36: Conexión null-modem. 67

Ilustración 37: MAX233..... 68

Ilustración 38: Adaptador RJ45-DB9. 68

Ilustración 39: Adaptador serie-USB. 69

Ilustración 40: Distribución dispositivos en el bus I2C..... 72

Ilustración 41: Diagrama de flujo programa telémetro por infrarrojos. 75

Ilustración 42: Diagramas de flujo de la función recepción/envío de datos de/a placa de control. 77

Ilustración 43: Diagrama de flujo de la función realizar operación. 78

Ilustración 44: Diagrama de flujo de la función realizar medida. 81

Ilustración 45: Diagrama de flujo programa placa de control. 83

Ilustración 46: Diagrama de flujo de la función recibir posición de la brújula. 86

Ilustración 47: Diagramas de flujo de la función enviar/recibir datos a/de el esclavo... 87

Ilustración 48: Función abrir y configurar puerto. 92

Ilustración 49: Función enviar cadena. 92

Ilustración 50: Función recibir cadena. 93

Ilustración 51: Función para obtener la orientación de la brújula. 94

Ilustración 52: Función cálculo código comprobación datos infrarrojos. 95

Ilustración 53: Función cálculo código de comprobación datos sonar. 96

Ilustración 54: Función cerrar puerto. 96

Ilustración 55: Diagrama de flujo del programa de prueba. 97

Ilustración 56: Curva DEC vs Lreal obtenida experimentalmente. 100

Ilustración 57: Valores DEC para L=30cm. 102

Ilustración 58: Valores DEC para L=60cm. 103

Ilustración 59: Valores DEC para L=80cm. 103

Ilustración 60: Realización de medidas con objetos en sensores 1,3 y 7. 106

Ilustración 61: Medidas realizadas con el telémetro de infrarrojos. 106

Ilustración 62: Medidas realizadas con la brújula digital. 110

Ilustración 63: Esquema de conexiones del sistema general. 111

Ilustración 64: Realización de medidas con sonar e IR. 112

ÍNDICE DE TABLAS

Tabla 1: Modos de operación del PIC16F876. 38

Tabla 2: Valor del registro SPBRG para transmisión asíncrona. 45

Tabla 3: Registros del micro del módulo CMPS03. 56

Tabla 4: Configuración de puerta de los medidores. 85

Tabla 5: Informe de errores. 91

CAPÍTULO 1.- INTRODUCCIÓN

1.1 Presentación y motivación

Una de las principales investigaciones en el mundo de la robótica móvil, está centrada en dotar a los robots de autonomía, que les permita desplazarse e interactuar con su entorno y adaptarse a cualquier modificación que se produzca en el mismo. Para ello es necesario implementar un sistema de navegación en el robot.

Para que un robot móvil pueda desplazarse desde un punto inicial hasta un punto final marcado, en un entorno desconocido, evitando los obstáculos que se encuentre en el camino, se requiere, entre otros, de sensores que permitan la realización de un mapa o modelo del entorno donde se sitúa el robot (sistema de visión artificial o de orientación por voz) y sensores de detección de obstáculos (telémetros laser, por infrarrojos, por ultrasonidos…) que le permitan seguir el camino sin chocar contra ningún objeto.

1.2 Objetivos

Para la realización de este proyecto se fijan varios objetivos. En primer lugar, desarrollar un telémetro por infrarrojos, por medio de los sensores Sharp GP2D02, determinando su rango de medida y permitiendo que el robot pueda detectar todos los objetos que se encuentren a su alrededor en un radio de distancias pequeño.

En segundo lugar, implementar el software necesario para obtener la orientación del robot, respecto al norte magnético, determinada por una brújula digital.

En tercer lugar, desarrollar una placa que permita la integración y el control de los telémetros de los sensores de detección de obstáculos situados en la base del robot, ultrasonidos e infrarrojos, y de una brújula digital. Esta placa permitirá realizar diferentes secuencias de disparo de los sensores, así como la obtención de los datos recogidos por los mismos. Desarrollar además el software y hardware necesario para establecer una comunicación bidireccional entre los diferentes dispositivos y la placa de control.

En el diseño de las placas siempre se tiene que tener presente su integración en el robot y, por problemas de espacio en el mismo, se tiene que tener especial cuidado con los parámetros dimensionales.

Por último, desarrollar un driver para que la placa de control, y en consecuencia todo el sistema, pueda ser manejado desde el PC del robot. De esta forma la función de la placa de control queda claramente definida: controlar el sistema sensorial y actuar como interfaz entre el PC y dicho sistema.

Se pretende tener un sistema lo más robusto posible y en el que se pueda identificar, de forma rápida y sencilla, cualquier fallo que se produzca durante el funcionamiento del mismo.

1.3 Los robots personales

Los robots personales son una nueva tecnología que se ha ido desarrollando, de manera muy rápida, durante los últimos años. El objetivo de este tipo de robots es interactuar con el ser humano y ayudarle en la realización de tareas rutinarias en el ámbito doméstico. La interacción hombre-robot es posible gracias a que se dota a los robots de gran cantidad de sensores y actuadores que les permiten adaptarse al entorno y tomar decisiones adecuadas respecto al mismo, de forma cada vez más semejante a la de los humanos.

Su incorporación a la vida cotidiana está condicionada a que éstos sean capaces de desenvolverse en el mundo de las personas sin tener que introducir en ese entorno modificaciones importantes.

El aspecto físico de un robot puede variar drásticamente en función de los componentes mecánicos y electrónicos incluidos en su configuración. Sin embargo, el aspecto más adecuado de un robot personal debe recordar la forma de una persona. La razón es doble, por un lado, la comunicación con las personas será mucho más natural si éstos disponen al menos de una tosca apariencia humana; por otra parte, permite al robot integrarse en el entorno humano, puesto que, el entorno físico de las personas, como por ejemplo, la vivienda, el coche, los centros de trabajo y ocio, están adaptados para su utilización por seres con piernas para moverse y brazos con manos para agarrarse, pulsar... Como ejemplo se pueden ver, en la Ilustración 1, diferentes robots que se han desarrollado en los últimos años.

De izquierda a derecha: robot Qrio de Sony[1], robot U-ROBO (ED-7270) del fabricante coreano ED[2].

Ilustración 1: Robots personales.

El sistema de locomoción debe ser ágil, fiable, seguro y autónomo, con materiales cuyo coste no sea muy elevado, pero sin renunciar a la garantía de la máquina.

Un robot tiene que tener integrado un sistema sensorial y otros elementos que le permita interpretar cualquier tipo de escena cotidiana, como, por ejemplo, detectar objetos o peligros y sortearlos, reconocimiento de caras... facultando al robot para desarrollar diversas habilidades o tareas.

Si tiene brazos u otros componentes extensibles, éstos deben de ser sensibles al mismo tiempo que fuertes, capaces de manipular cualquier tipo de objeto sin romperlo o deteriorarlo.

Para la realización de tareas debe tener un "cerebro" capaz de aprender por sí solo los conceptos y los procedimientos necesarios, así como mantener el entendimiento con su entorno de manera abierta y flexible.

Una tarea esencial que un robot móvil personal debe ser capaz de realizar de forma autónoma es moverse por su entorno, lo que implica dos habilidades básicas. En

[1] http://www.sony.com.pa/qrio/tecnologia.html
[2] http://es.engadget.com/2006/11/02/el-robot-domestico-u-robo-de-ed-con-rfid/

primer lugar ser capaz de moverse sin chocar con objetos, personas... que encuentre a su paso. En segundo lugar, conocer su posición en el entorno, o lo que es lo mismo, no perderse o tener la capacidad de auto-localización. El grado de autonomía depende en gran medida de la capacidad del robot para abstraer el entorno y convertir esa información en órdenes, de tal modo, que aplicadas sobre los actuadores del sistema de locomoción, garantice la realización eficaz de su tarea.

Otras habilidades pueden ser:

- Comunicarse mediante el lenguaje hablado. En esta tarea se han realizado importantes progresos, no sólo en el reconocimiento de las palabras sino también en la posibilidad de reconocer el contexto del discurso y así captar su significado.
- Responder a diferentes estímulos. Para ello se dota al robot de los elementos necesarios, como cámaras o sensores, que permitan desarrollar esta habilidad.
- Vigilancia: puede actuar como un guarda, detectando intrusos, fuego, pérdidas de agua y posibles inundaciones...
- Información: usando servicios web puede suministrar información sobre el tiempo, el tráfico en las carreteras, noticias de última hora...
- Tareas domésticas: puede limpiar el suelo, llevar objetos de un sitio a otro, encendido mediante mando a distancia de diferentes electrodomésticos...
- Ofrecer entretenimiento y actividades de ocio como jugar a videojuegos o escuchar música.
- Ayudar a organizar actividades, recordar reuniones o encuentros.
- Ayudar a personas discapacitadas o dependientes.
- Realizar diversas aplicaciones domóticas como subir persianas, control del alumbrado, regular la calefacción en función de la temperatura exterior, regular la luminosidad de la estancia...

1.4 *Maggie*, robot donde se integra el proyecto

El robot móvil personal donde se integra el proyecto se llama *Maggie*. *Maggie* es un robot móvil porque puede desplazarse por el entorno con las dos ruedas motrices de su base, además puede mover los brazos (un grado de libertad cada uno, pueden girar alrededor de su eje), la cabeza (dos grados de libertad, movimiento vertical y horizontal) y los párpados (un grado de libertad cada uno, pueden moverse verticalmente).

Como se puede observar en la Ilustración 2, su apariencia amigable y simpática le permite interactuar con adultos y niños, sin que, sobre todo estos últimos, lo perciban como un peligro o le tengan miedo, cumpliendo de este modo una de las características aconsejables que debe tener este tipo de robot. *Maggie* está siendo desarrollado en el Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid, por el grupo "Robotics Lab".

Ilustración 2: Robot móvil personal *Maggie*.

Desde el punto de vista de la construcción, el robot se puede dividir en dos partes distintas: la base y el cuerpo.

1.4.1 Base del robot

Se trata de la parte fundamental del robot, gracias a la base, *Maggie*, puede realizar todas las funciones que dispone.

La base tiene un diámetro de 40cm y se divide en dos puertas, la puerta delantera o frontal y la puerta trasera.

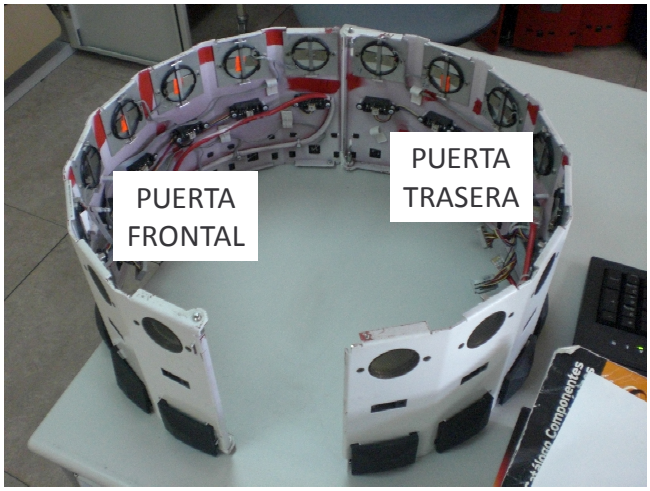


Ilustración 3. Puertas frontal y trasera de la base del robot.

En la base se encuentra el sistema locomotor, el sistema de alimentaciones y el sistema de emergencias [1].

En cuanto al sistema de alimentación, *Maggie* está alimentada con dos baterías de +12V y de 12A/hora. El voltaje de estas baterías está regulado por tres conversores DC-DC (direct current - direct current), los cuales fijan las tensiones del robot en +5V y +12V. La alimentación del sistema locomotor es independiente del resto de alimentaciones del robot.

El sistema de alimentación para los drivers se encuentra en la placa de emergencias, que es independiente del resto de alimentaciones.

El sistema locomotor consta de dos motores que transmiten mediante correas el movimiento a las ruedas.

El sistema de emergencias es el encargado de la parada de los seis motores que controlan el movimiento de *Maggie* (base, brazos y cabeza) en cualquier momento siempre que lo requiera el usuario. Estas paradas se llevan a cabo mediante una seta de emergencia activada por radiofrecuencia.

1.4.2 Cuerpo del robot

Sobre la base y solidaria a ella (sin ningún grado de libertad), se ha construido una estructura de acuerdo con el objetivo para el que se ha diseñado este robot: su interacción con los humanos.

Maggie mide 140cm y tiene una cámara integrada en la cabeza y otra en el tronco y un Tablet-PC en el pecho. La integración de estos equipos permite al robot un total conocimiento del entorno en el que se mueve (extrayendo información a partir de las imágenes de la cámara del tronco) y la interacción con las personas, por ejemplo, la cámara de la cabeza, que le permite un reconocimiento de caras y conocer de esta forma a la persona que tiene delante. Otro ejemplo es que a través del Tablet-PC, *Maggie* puede representar gestos mediante caras dibujadas, de esta forma puede expresar sorpresa, enfado, risa…

1.4.3 Sistema sensorial

El robot tiene distribuidos a lo largo de toda su estructura diversos sensores, que permitirán su interacción con el entorno y le darán cierta autonomía para moverse por el mismo.

La estructura de la base (Ilustración 3 e Ilustración 4) es un hexadecágono, en la que hay integrados diferentes sensores para la detección de obstáculos, en concreto, 16 sensores de ultrasonidos (a 20.5cm de altura), 16 sensores de infrarrojos (a 14.5cm de altura) y 16 sensores de choque o bumpers (a 4cm de altura). La distancia entre un sensor y el adyacente es de 8 cm.

- Sensor láser PLS[1] SICK:

La colocación del mismo en la estructura del robot se muestra en la siguiente ilustración. Se utiliza como parte del sistema de navegación y sirve para determinar, en cualquier instante, la posición del robot dentro de un entorno conocido.

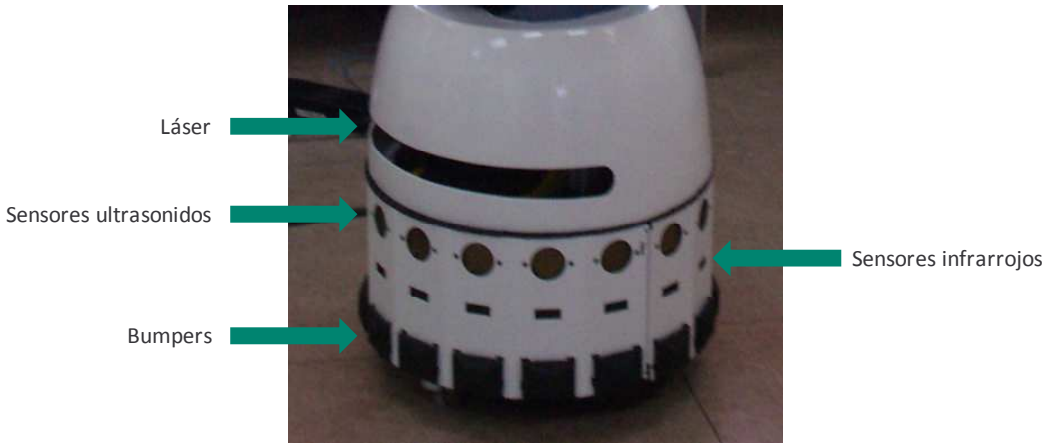


Ilustración 4: Distribución de los sensores en la base de Maggie.

En los brazos y tronco del robot se encuentran distribuidos sensores de tacto. Se utilizan para la localización y reconocimiento de objetos, así como para controlar la fuerza ejercida sobre los mismos, esto le permite su interacción con el entorno.

Integrada en la placa de control del sistema sensorial de la base del robot, hay una brújula digital que va a permitir la orientación del robot (puesto que se va a conocer su posición (respecto al norte magnético)) y el desplazamiento del mismo según unas coordenadas determinadas.

1.5 Evolución del mercado de robots sociales

Aunque está en vías de consolidación, en términos generales, el mercado de los robots móviles personales está inmaduro y todavía no se ha abierto al gran público en general. Esto se debe en parte a la fragilidad de los resultados (en comparación con las expectativas) y a las limitaciones en autonomía que tienen los robots conseguidos hasta

[1] Pacific Laser System

la fecha. De hecho, el grado de autonomía y de fiabilidad que se pueda conseguir en los robots será una cuestión determinante en la evolución futura de este mercado.

En los medios de comunicación aparecen cada vez más noticias de robots con aspecto humano realizando alguna tarea que requiere cierta habilidad, como por ejemplo subir una escalera, abrir puertas o transportar personas. Todavía queda un largo camino para que estas habilidades dejen de ser noticia y se puedan comprar en la tienda de la esquina robots parecidos a los de las películas de ciencia ficción, pero se espera que con el paso de los años el mercado crezca de forma rápida.

Según la International Federation of Robotics (IFR), en la actualidad, hay alrededor de cinco millones de robots personales en el mundo [2]. El informe World Robotics 2008, estudio que se publica cada año y que es desarrollado por el IFR Estatistical Department, estima que habrá 7.2 millones de este tipo de robots a finales del año 2010 [3], lo que supone una venta de 2.2 millones de robots en los próximos dos años.

CAPÍTULO 2.- ARQUITECTURA HARDWARE

En este capítulo se va a detallar el desarrollo de la arquitectura hardware del sistema de control sensorial, describiendo los diferentes dispositivos integrados en la misma. En primer lugar se expondrá una visión general de todo el sistema, presentando los niveles que componen la arquitectura y posteriormente se particularizará, describiendo cada dispositivo, así como los componentes que lo integran.

2.1 Descripción y esquema general

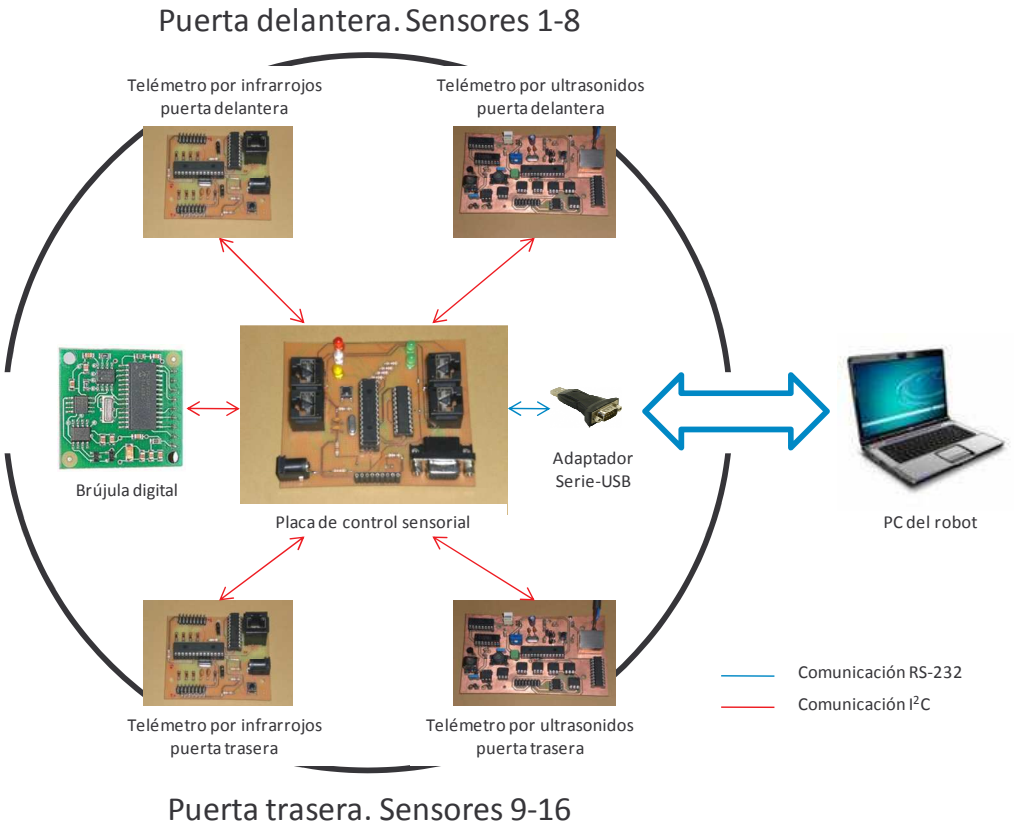


Ilustración 5. Esquema general del sistema.

Como ya se ha comentado en el capítulo anterior, la base del robot *Maggie* es un hexadecágono, en el que, en cada uno de sus lados, se encuentran integrados 16 sensores de ultrasonidos, 16 sensores de infrarrojos y 16 sensores de choque o bumpers. Distribuidos en dos grupos de ocho, un grupo en la puerta delantera o frontal (sensores del 1 al 8) y el otro en la puerta trasera (sensores del 9 al 16). Puertas que conforman la base del robot.

Los sensores de ultrasonidos y los sensores de infrarrojos se utilizan para detectar objetos u obstáculos, indicando la distancia a la que se encuentran los mismos, para evitar que el robot pueda chocar contra ellos.

En este proyecto se ha diseñado e implementado un telémetro o medidor de distancias mediante sensores de infrarrojos (Sharp-GP2D02), desarrollando tanto la

parte hardware como la parte software del mismo. También se ha integrado el "módulo CMPS03" que contiene sensores magnéticos y funciona como brújula digital y se ha integrado un telémetro o medidor de distancias mediante sensores de ultrasonidos (Serie 600 de Polaroid) (desarrollado, este último, en otro proyecto).

Dentro de la parte hardware del medidor de distancias de los sensores de infrarrojos, al estar formada la base del robot por dos puertas, se han desarrollado dos placas, una para los sensores colocados en la puerta delantera y otra para los sensores colocados en la puerta trasera. Teniendo siempre en cuenta en su diseño, los requerimientos dimensionales para su integración en la base de *Maggie*.

En cuanto a la interacción con otros dispositivos, los medidores de distancias de los sensores utilizados en la detección de obstáculos (infrarrojos y sonar), han sido diseñados para permitir su comunicación serie. La comunicación con cualquier ordenador se establece mediante el protocolo RS-232[1] y la comunicación con la placa de control se establece mediante el protocolo I^2C[2]. Esto permite integrarlos fácilmente en el sistema y su control desde cualquier ordenador para realizar pruebas de funcionamiento de los mismos.

Según se puede observar en la imagen de inicio de este capítulo, la parte central de este proyecto es el diseño y desarrollo de una placa de control sensorial, que permite el control de los medidores de distancias y de la brújula digital y actúa como interfaz entre el PC del robot y el sistema sensorial del mismo. Esto permitirá desplazarse y orientarse al robot respecto a los puntos cardinales, dotando de más autonomía a su movilidad.

El control se efectuaría del siguiente modo: el PC del robot, se comunica con la placa de control, indicándole los sensores que se disparan y la secuencia de disparo de los mismos. Una vez recogidos estos datos, el sistema funciona como un sistema maestro-esclavo. Donde el maestro es la placa de control sensorial y los esclavos son los telémetros y la brújula digital. Se obtienen los resultados y se envían al PC del robot. Como ya se ha mencionado anteriormente la comunicación entre todas las placas se efectúa siguiendo el protocolo I^2C.

[1] Recommended Standard 232→ Comunicación serie asíncrona
[2] Inter-Integrated Circuit → Comunicación serie síncrona. Desarrollado por Philips Semiconductors.

2.2 Descripción hardware del telémetro por infrarrojos

En este apartado se van a describir todos los componentes que se han utilizado en el diseño y desarrollo del medidor de distancias mediante los sensores de infrarrojos Sharp GP2D02.

Como se ha mencionado anteriormente, se han diseñado dos placas iguales, que irán integradas en la base del robot, una en cada puerta, y a la que, a cada una, se conectan ocho sensores. Las dimensiones de cada una de las placas son: 63x74mm. En el ANEXO B se puede ver el esquemático y el layout de las placas, diseñados ambos con el programa ORCAD v10.0.

En la ilustración, se puede observar una de las placas:

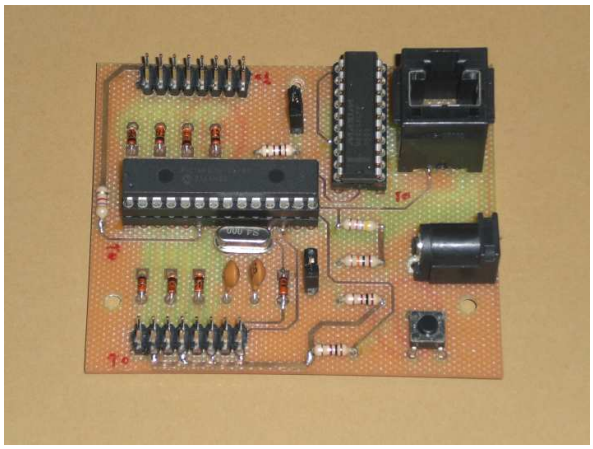


Ilustración 6: Medidor sensores infrarrojos.

2.2.1 Sensor Sharp GP2D02

Un sensor es un dispositivo capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas. Las variables de instrumentación dependen del tipo de sensor y pueden ser, por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, pH… La magnitud eléctrica obtenida puede ser una variación de resistencia eléctrica, de capacidad eléctrica, de tensión eléctrica, de corriente eléctrica…[4]

Para este proyecto no se ha podido elegir el tipo de sensores de detección de obstáculos a utilizar, porque la base del robot es una base comercial denominada Maguellan Pro, en la que ya vienen integrados los sensores de ultrasonidos (Serie 600 de Polaroid) y los sensores de infrarrojos (Sharp GP2D02).

Los sensores de infrarrojos Sharp GP2DXX son una familia de sensores utilizados para la detección de objetos. Se clasifican dentro de los sensores reflexivos. Este tipo de sensores presentan una cara frontal (Ilustración 7) en la que se encuentra tanto el LED emisor como el fototransistor receptor. Debido a esta configuración el sensor detecta la radiación proveniente del reflejo de la luz infrarroja emitida por el LED al chocar contra un objeto.

Ilustración 7: Sharp GP2D02.

En cuanto a las características eléctricas, en particular, el sensor Sharp GP2D02 es un sensor digital que utiliza una línea de entrada (Vin) y otra de salida (Vout) para comunicarse con el microprocesador[1] principal. El dispositivo se alimenta poniendo a +5V el pin VCC y GND a 0V (masa). El pin Vout es la salida serie digital de datos con lógica positiva y niveles TTL[2].

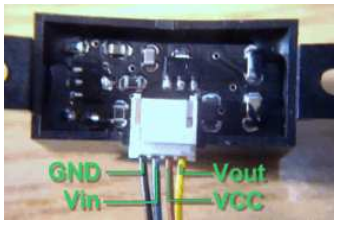


Ilustración 8: Terminales del Sharp GP2D02.

[1] PIC16F876
[2] Transistor-Transistor Logic → Niveles lógicos definidos por el rango de tensión 0-5V

Vin es la entrada con la que se comanda el funcionamiento del sensor, hay que tener muy en cuenta una característica de este pin y es que su salida es a drenador abierto y poner esta entrada a una salida TTL o CMOS[1] provocaría la destrucción de la misma. Sólo acepta niveles bajos y por lo tanto para acoplarla a la salida del PIC16F876 es necesario poner un diodo. En la ilustración siguiente se muestra el conexionado entre los terminales del sensor y el microcontrolador [5]:

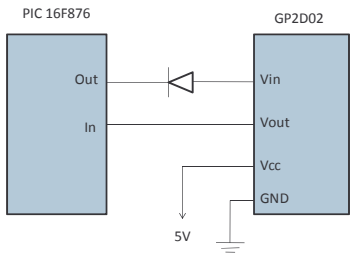


Ilustración 9: Conexión PIC16F876 a GP2D02.

Algunos datos que hay que tener presentes antes de utilizar el sensor en cualquier sistema son los siguientes:

- Rango de medida: 10 - 80cm (determinado por el fabricante).
- Temperatura de funcionamiento: -10 – 60ºC.
- Alimentación: 4,4 - 7V.
- Consumo (activo/reposo): 35mA/2mA.
- Insensibilidad al color.
- Sensibilidad a la luminiscencia.
- Dimensiones: 37mm de longitud y 14mm de altura, el emisor y el receptor se encuentran distanciados 20mm.

2.2.1.1 Principio de funcionamiento del sensor:

La idea básica de funcionamiento del Sharp GP2D02 es la emisión de un pulso de luz infrarroja (longitud de onda mayor a 0,7µm), que viaja a través del campo de visión y tropieza con un objeto o continúa. En caso de que no exista un objeto, la luz nunca es reflejada y la lectura muestra que no hay objeto. Si la luz es reflejada por un objeto, vuelve al detector y crea un triángulo entre el punto de reflexión, el emisor y el detector. Los ángulos de este triángulo están relacionados con la distancia al objeto [6].

[1] Complementary Metal Oxide Semiconductor → Utilización conjunta de transistores de tipo pMOS y tipo nMOS

Cuanto más cerca esté el objeto del sensor el ángulo será más grande y el triángulo que se forma más pequeño.

El dispositivo emite luz infrarroja por medio de un led emisor de infrarrojos, esta luz pasa a través de una lente que concentra los rayos de luz formando un único rayo lo más concentrado posible para así mejorar la directividad del sensor, la luz va recta hacia delante y cuando encuentra un obstáculo reflectante rebota y retorna con cierto ángulo de inclinación dependiendo de la distancia, la luz que retorna es concentrada por otra lente y así todos los rayos de luz inciden en un único punto del sensor. Dependiendo del ángulo de recepción de la luz incidirá, ésta, en un punto u otro del sensor pudiendo de esta manera obtener un valor proporcional al ángulo de recepción del haz de luz [5].

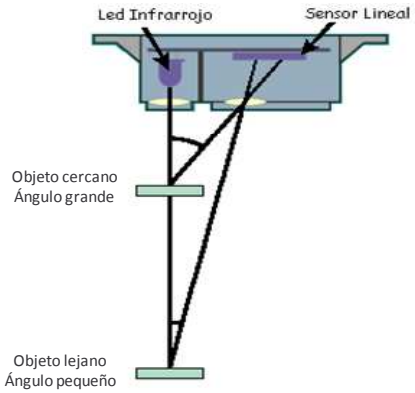


Ilustración 10: Funcionamiento Sharp GP2D02.

El sensor proporciona un valor de 8 bits (0-255) relacionado con el ángulo que forma la luz reflejada al chocar contra el objeto. Se tiene que tener presente que la lectura proporcionada por el sensor es sensible a la luminosidad del entorno perjudicando las medidas y dando lugar a errores, pudiendo ser necesario la incorporación de circuitos de filtrado en términos de longitud de onda, por lo que, para evitar esto, será importante controlar la luz en el entorno en que se mueva el robot. Otro aspecto fundamental a tener en cuenta es la reflectividad del objeto, la lectura del sensor también variará en función del tipo de superficie del objeto: acabado superficial, rugosidad, porosidad...

La reflectividad puede ser definida como la fracción del flujo de radiación incidente que es reflejado en la superficie de un objeto [13] y está relacionada con el coeficiente de refracción, cuanto mayor sea éste, mayor será la reflectividad.

La luz infrarroja es una radiación del espectro luminoso, que tiene mayor longitud de onda y menor frecuencia que el espectro visible y se encuentra más allá del rojo visible, como se puede ver en la Ilustración 11. No es perceptible por el ojo humano y en este caso se utiliza para detectar objetos aunque tiene otras muchas aplicaciones: detectar imperfecciones en superficies, detectar diferencias de temperatura…

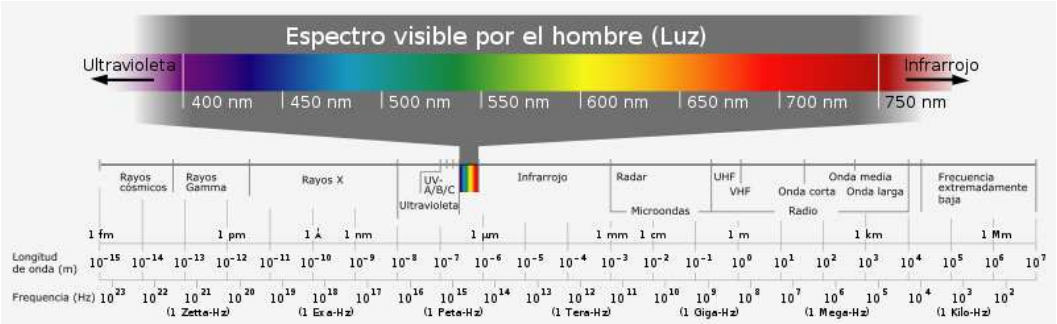


Ilustración 11: Espectro electromagnético.

2.2.1.2 Protocolo para la obtención de medidas:

Una vez que se tienen conectados los terminales del sensor al microcontrolador, sólo queda "disparar" el sensor y leer el resultado obtenido. En las hojas de características[1] del sensor se describe la forma de conseguirlo, que se detalla a continuación.

La entrada Vin debe estar inicialmente a nivel lógico '1' para mantener el sensor inactivo. En cuanto la señal baja a nivel lógico '0', el sensor comienza la operación de medida. El tiempo que tarda el sensor en enviar la luz infrarroja y obtener la medida es como máximo de 70ms, por lo que es necesario mantener a nivel lógico '0' la señal durante este tiempo. Para obtener el valor de la medida recogida por el sensor, la señal pasa a nivel lógico '1' durante un tiempo inferior a 0.2 ms (recomendado 0.1ms) e inmediatamente hay que volver a ponerla a nivel lógico '0'. En este flanco de bajada, el

[1] Ver ANEXO A

sensor pone en la salida Vout el bit más significativo (MSB) del byte correspondiente a la medida. Si repetimos esta señal cuadrada 7 veces más, obtenemos el byte con la medida. El último bit dado es el menos significativo (LSB). Una vez recibido el último bit (octavo flanco de bajada), hay que volver a poner la entrada Vin a nivel lógico '1', manteniéndola así hasta que se necesite realizar otra nueva medida. Si se quieren realizar medidas consecutivas, es necesario esperar con la entrada Vin a nivel lógico '1' al menos 1,5ms antes de la siguiente medida.

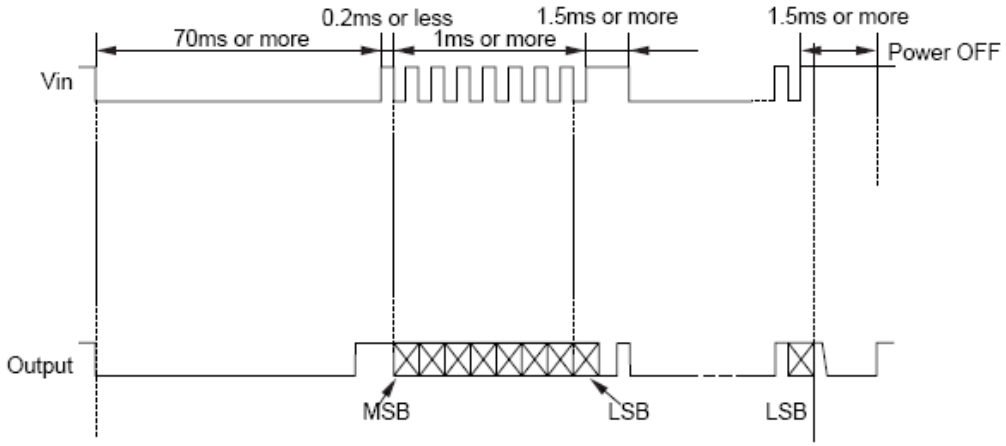


Ilustración 12: Vin y Vout para conseguir una medida.

El sensor tarda aproximadamente 70ms en obtener una medida, aunque este valor puede variar en función de la reflectividad de la superficie del objeto, la distancia y otros factores. Cuando se obtiene la medida, la salida Vout se pone a nivel lógico ‘1’, teniendo en cuenta esto, se puede reducir el tiempo de realización de una medida.

El tiempo de muestreo óptimo de los sensores es:

$$T_{muestreo} = 70ms + (0.1ms \times 2 \times 8) + 1.5ms = \ 73.1\ ms$$

$$f_{muestreo} = \frac{1}{T_{muestreo}} = 13{,}7Hz$$

2.2.1.3 Calibración del sensor:

La relación entre el valor proporcionado por el sensor (en adelante DEC) y la distancia a la que se encuentra el objeto (en adelante L) se puede observar en la Ilustración 13.

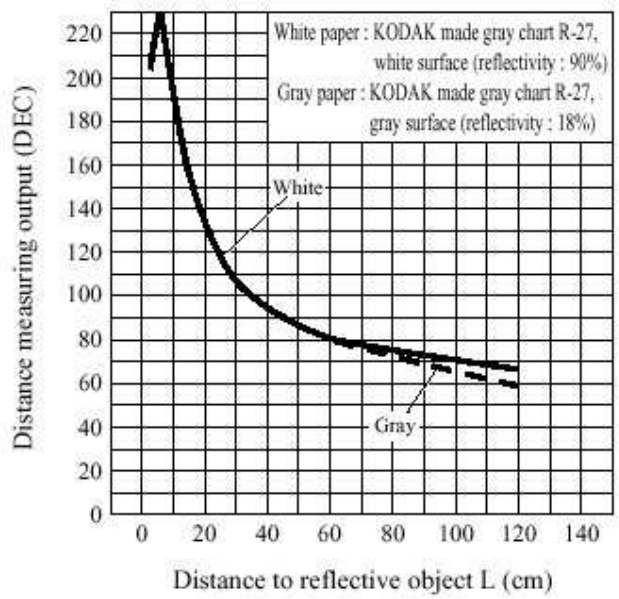


Ilustración 13: Relación entre la distancia dada por el sensor (DEC) y la distancia real (L).

Se puede comprobar en la ilustración anterior como influye la reflectividad del objeto sobre DEC y sobre L, según la hoja de características del sensor proporcionada por el fabricante.

La relación entre DEC y L no es lineal, cuando el objeto está cercano al sensor (distancia inferior a 7cm) el valor DEC aumenta hasta alcanzar un máximo, a partir de este punto de inflexión el valor DEC disminuye de forma inversamente proporcional a la distancia. La relación entre DEC y L queda parametrizada por medio de la siguiente fórmula [12]:

$$L(cm) = \frac{Kg}{(DEC - Ko)} * 100$$

Kg: Constante que determina la forma de la curva.

Ko: Constante que determina la posición de la curva (más arriba o más abajo en el plano).

En el capítulo 4 “Resultados experimentales”, se estudiará en profundidad el comportamiento del sensor, comprobando todos los aspectos detallados anteriormente así como la realización de medidas y el cálculo de errores.

2.2.2 Microcontrolador de los sensores IR: PIC16F876

Un microcontrolador es un circuito integrado, que consta de una CPU[1], memoria interna y unidades Entrada/Salida.

La elección de este micro, y no otros, ha sido por utilizar los recursos disponibles, puesto que en el laboratorio donde se ha desarrollado el proyecto ya existían dos programadores de este tipo de micro y si se utilizaba otro tipo habría que comprar el programador necesario.

El PIC16F876 se escogió para el telémetro por infrarrojos, puesto que tiene los puertos E/S necesarios y una capacidad de memoria adecuada. También se podrían haber utilizado otros micros, como por ejemplo: BasicX-24, Basic Stamp 2, 8051 de Intel…

El microcontrolador es el encargado de controlar todo el proceso de adquisición de medidas del sensor de infrarrojos. Genera la señal de disparo de los sensores (pulsos en el terminal Vin) y lee el resultado obtenido de los mismos (pulsos en el terminal Vout). Es un micro desarrollado y fabricado por Microchip Technology Inc[2].

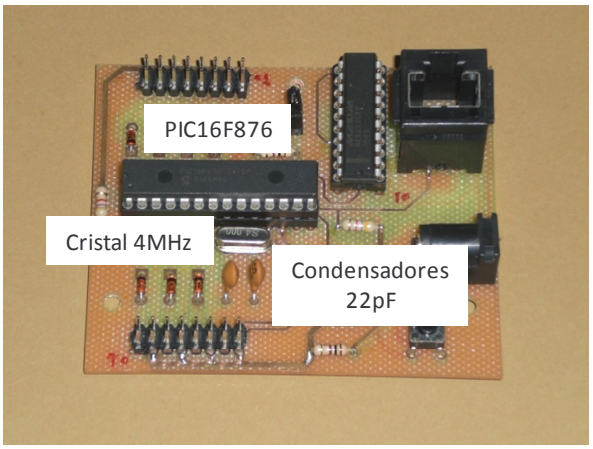


Ilustración 14: PIC16F876 y circuito oscilador.

[1] En inglés: Central Processing Unit.
[2] http://www.microchip.com

Como se puede observar, tiene 28 pines y encapsulado PDIP[1]. Los siguientes apartados se han redactado teniendo en cuenta el manual del PIC [9].

Entre sus características principales cabe destacar:

- Micro de 8 bits (bus de datos).
- Memoria CMOS FLASH[2] de 8Kx14bits (corresponde a la F del nombre).
- Memoria SRAM de 368 bytes.
- Memoria EEPROM de 256 bytes.
- 22 entradas/salidas digitales distribuidas en 3 puertos (A, B y C) y 4 entradas analógicas disponibles en el puerto A.
- 3 contadores/temporizadores, dos de 8 bits y uno de 16 bits.
- Convertidor A/D[3] → de 10 bits/5canales.
- 2 CCP → Módulos Capture/Compare/PWM.
- Un puerto serie síncrono → Puede ser configurado como Serial Peripheral Interface (SPI) o como Inter-Integrated Circuit (I²C).
- Un Universal Synchronous Asynchronous Receiver Transmitter (USART).
- 14 Interrupciones.
- 35 Instrucciones.
- Voltaje de funcionamiento (Vdd): 2 - 5,5V.

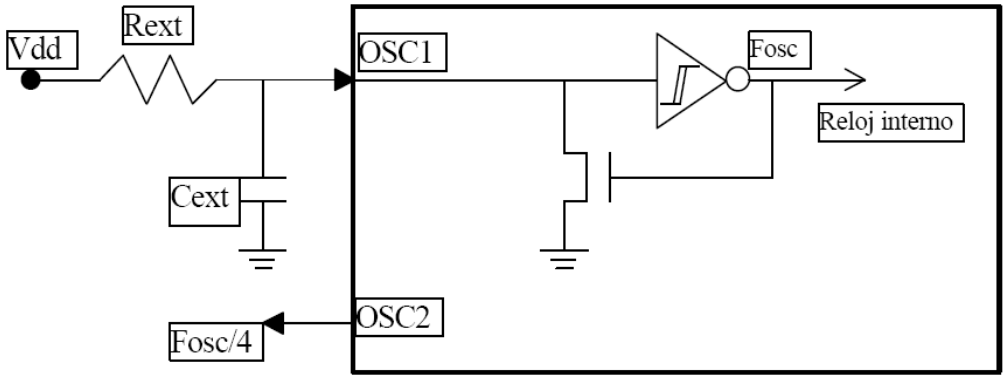
2.2.2.1 Oscilador:

El oscilador va a generar los pulsos que sincronizan todas las operaciones internas del micro, así como el incremento/decremento de la cuenta de los timer. La velocidad de ejecución de las instrucciones del programa (grabado en la memoria EEPROM del micro) está relacionada con la frecuencia del oscilador.

Para la realización de este proyecto se ha utilizado un reloj externo al micro. Se permiten las siguientes configuraciones:

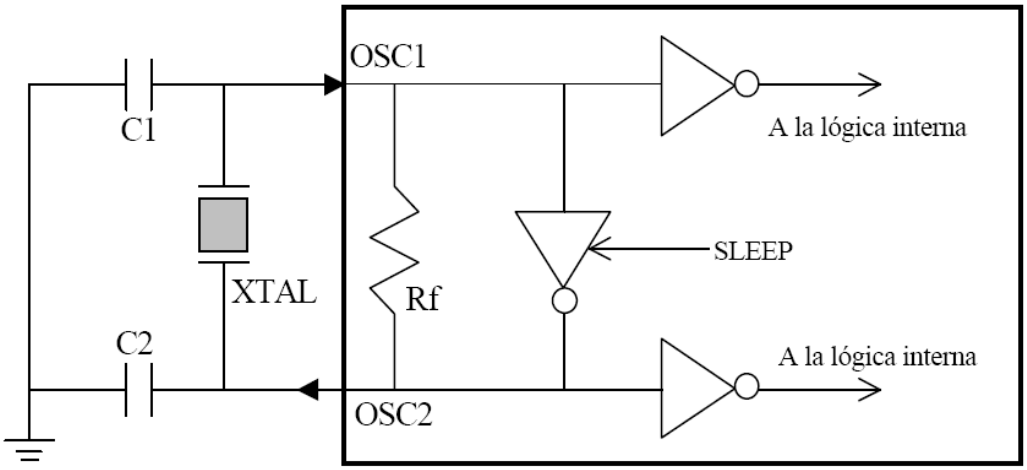
[1] Plastic Dual In Line Package
[2] Permite que múltiples posiciones de memoria sean escritas o borradas en una misma operación de programación mediante impulsos eléctricos.
[3] Analógico/Digital

- Circuito RC externo conectado al pin 9 del micro.



Se recomienda, esta configuración, cuando la aplicación no requiera una gran precisión en la medición de tiempos.

- Cristal de cuarzo externo conectado entre los pines 9 y 10 del micro.



Se recomienda esta configuración debido a su gran estabilidad de frecuencia. Con esta configuración existen tres modos posibles de operación: LP, XT y HS.

Modo	Frecuencia cristal típica	Condensadores recomendados	
		C1	C2
LP	32KHz	68-100 pF	68-100 pF
	200 KHz	15-30 pF	15-30 pF
XT	100 KHz	68-150 pF	150-200 pF
	2MHz	15-30 pF	15-30 pF
	4MHz	15-30 pF	15-30 pF
HS	8MHz	15-30 pF	15-30 pF
	10MHz	15-30 pF	15-30 pF
	20MHz	15-30 pF	15-30 pF

Tabla 1: Modos de operación del PIC16F876.

En el proyecto se ha utilizado esta segunda configuración, utilizando un cristal de cuarzo de 4MHz y 2 condensadores de 22pF cada uno.

2.2.2.2 Circuito de Reset:

Como reset se utiliza un circuito que se conecta al pin 1 del micro. El circuito está formado por una resistencia conectada a la alimentación (Vdd) del micro y un pulsador conectado a tierra, tal y como se puede observar en la siguiente ilustración:

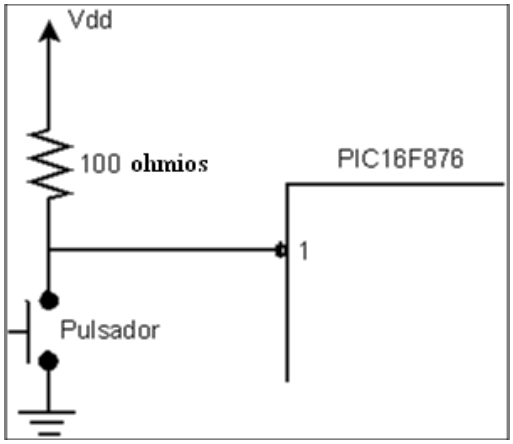


Ilustración 15: Circuito de reset del PIC16F876.

De esta forma cuando se accione el pulsador, el pin 1 se pondrá a nivel lógico '0' y se reseteará el micro.

2.2.2.3 Memoria de datos SRAM:

La memoria de datos consta de dos áreas mezcladas y destinadas a distintas funciones:

- Registros de Función Especial (SFR).
- Registros de Propósito General (GPR).

Los SFR son posiciones de memoria de 8 bits asociadas específicamente a los diferentes periféricos y funciones de configuración del PIC y tienen un nombre específico, asociado con su función. Mientras que los GPR son posiciones de memoria RAM de uso general.

2.2.2.4 Periféricos integrados en el micro y utilizados en este proyecto:

2.2.2.4.1 Puertos I/O:

Los puertos del micro son totalmente programables, es decir, sus líneas pueden ser configuradas para trabajar como entradas o como salidas a elección del programador.

- Puerto A: tiene 6 líneas bidireccionales. Los registros asociados al puerto son:
  - PORTA (05h[1])

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000

Los 6 bits menos significativos del registro se corresponden con la línea física homónima del puerto. Realizar una lectura de este registro implica leer el estado de todos los pines del puerto. Toda escritura en este registro implica que el estado de los pines se lee, luego se modifica y posteriormente se escribe al latch[2] de datos del puerto.

[1] Dirección en hexadecimal de la posición de memoria del registro en la SRAM
[2] Circuito electrónico usado para almacenar información en sistemas lógicos asíncronos

- TRISA (85h)

Cada bit de este registro configura la línea física homónima como entrada o como salida. Si el bit es igual a uno, la línea física será configurada como entrada y si el bit es igual a cero, la línea física será configurada como salida. Al realizar un reset del micro todos los bits del registro se ponen a nivel lógico '1' (entradas).

- ADCON1 (9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Antes de utilizar el puerto A hay que configurar los pines como entradas/salidas analógicas o como entradas/salidas digitales. Como para el proyecto se necesita esta segunda configuración, hay que escribir el valor binario '0110' en los cuatro bits menos significativos del registro. Las restantes configuraciones se pueden ver en el manual del PIC.

- Puerto B: tiene 8 líneas bidireccionales. Los registros asociados al puerto son:

  - PORTB (06h,106h)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu

Los 8 bits del registro se corresponden con la línea física homónima del puerto. Realizar una lectura de este registro implica leer el estado de todos los pines del puerto. Toda escritura en este registro implica que el estado de los pines se lee, luego se modifica y posteriormente se escribe al latch de datos del puerto.

- TRISB (86h,186h)

Cada bit de este registro configura la línea física homónima como entrada o como salida. Si el bit es igual a uno, la línea física será configurada como entrada y si el bit es igual a cero, la línea física será configurada como salida. Al realizar un reset del micro todos los bits del registro se ponen a nivel lógico '1' (entradas).

- OPTION_REG (81h,181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

El bit más significativo de este registro (bit 7) sirve para acoplar/desacoplar las resistencias de pull-up asociadas a cada pin del puerto B. Para conectar estas resistencias se debe colocar el bit a nivel lógico '0' y para desconectarlas se debe poner el bit a nivel lógico '1'. Las resistencias son automáticamente desacopladas cuando se configura el pin correspondiente como salida. En el proyecto se van a configurar como inicialmente acopladas.

- Puerto C: tiene 8 líneas bidireccionales. Los registros asociados al puerto son:

  - PORTC (07h)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu

Los 8 bits del registro se corresponden con la línea física homónima del puerto. Realizar una lectura de este registro implica leer el estado de todos los pines del puerto. Toda escritura en este registro implica que el estado de los pines se lee, luego se modifica y posteriormente se escribe al latch de datos del puerto.

  - TRISC (87h)

Cada bit de este registro configura la línea física homónima como entrada o como salida. Si el bit es igual a uno, la línea física será configurada como entrada y si el bit es igual a cero, la línea física será configurada como salida. Al realizar un reset del micro todos los bits del registro se ponen a nivel lógico '1' (entradas).

2.2.2.4.2 Timer 0:

El Timer 0 es un temporizador/contador de 8 bits (00h – FFh). El modo temporizador se selecciona poniendo a nivel lógico '0' el bit 5 del registro OPTION_REG (T0CS). El registro asociado a este Timer es el TMR0 (01h), éste se incrementará en una unidad a una frecuencia que dependerá de la frecuencia del

oscilador externo y una proporción determinada por un pre-escalador. Esta proporción se selecciona mediante los tres bits menos significativos del registro OPTION_REG. En el manual del PIC se explica con más detalle cuáles son las proporciones seleccionables. Para asociar el pre-escalador al Timer 0 hay que poner a nivel lógico '0' el bit 3 del registro OPTION_REG (PSA).

Para este proyecto se utiliza un oscilador de cristal de 4MHz y una proporción de pre-escalador 1/64, luego la frecuencia de incremento será:

$$fincremento = \frac{foscilador}{4} \times \frac{1}{64} = \frac{4MHz}{256} = 15{,}625kHz$$

El período de incremento (inversa de la frecuencia) será:

$$Tincremento = \frac{1}{fincremento} = \frac{1}{15{,}625kHz} = 64\mu s$$

Como se trata de un temporizador de 8 bits, el tiempo que tarda en incrementar desde 00h hasta FFh (256 unidades) es:

$$Ttotal = 64\mu s \times 256 = 0{,}016384s$$

Cuando el Timer finaliza la cuenta, es decir, cuando pasa del valor FFh al valor 00h se produce una interrupción, se pone a nivel lógico '1' el bit T0IF del registro INTCON (0Bh). Este bit se consulta para determinar el momento de fin de la cuenta del Timer.

El valor inicial del registro TMR0 es configurable, se puede escribir un valor de precarga inicial en este registro y empezar el incremento a partir de ese valor.

El Timer 0 se utiliza en el proyecto para temporizar los intervalos de la entrada Vin del sensor para obtener la medida. Estos intervalos de tiempo son:

- 70ms: tiempo de espera hasta que la medida está realizada. Valor de precarga del TMR0:

$$\frac{Tpulso}{Ttotal} = \frac{70ms}{0{,}016384} = 4{,}2724 \Longrightarrow Precarga = 256 \times (1 - 0{,}2724) = 186$$

Hay que esperar que se produzcan 4 interrupciones del Timer (fin de cuenta) y fijar en el registro TMR0 el valor de precarga establecido.

- 0,1ms: ancho de los 8 pulsos, en estado alto, para obtener el valor de la medida del sensor. Valor de precarga del TMR0:

$$\frac{Tpulso}{Ttotal} = \frac{0{,}1ms}{0{,}016384} = 0{,}0061 \Longrightarrow Precarga = 256 \times (1 - 0{,}0061) = 254$$

- 1,5ms: tiempo de espera entre medidas consecutivas. Valor de precarga del TMR0:

$$\frac{Tpulso}{Ttotal} = \frac{1{,}5ms}{0{,}016384} = 0{,}0915 \Longrightarrow Precarga = 256 \times (1 - 0{,}0915) = 232$$

2.2.2.4.3 El puerto serie USART:

La USART puede ser configurada para operar en tres modos: modo asíncrono (full-duplex[1]), modo síncrono–maestro (half-dúplex[2]), modo síncrono–esclavo (half-dúplex).

[1] Los datos se pueden enviar simultánea y bidireccionalmente.
[2] La línea de datos es bidireccional, pero no se pueden transmitir los datos simultáneamente

Para este proyecto se utiliza el modo asíncrono. Dicho modo se selecciona poniendo a nivel lógico '0' el bit SYNC del registro TXSTA (98h) y poniendo a nivel lógico '1' el bit SPEN del registro RCSTA. En el Capítulo 3 "IMPLEMENTACIÓN DEL SOFTWARE" se puede ver como se forman las tramas de datos según establece el protocolo serie asíncrono RS-232.

- Circuito de muestreo:

Cuando se recibe el dato en el pin 18 (RC7/RX/DT), éste es muestreado tres veces para poder decidir mediante un circuito de mayoría, si se trata de un nivel alto o un nivel bajo.

- Baud Rate Generator (BRG):

Es un contador/divisor de frecuencia de 8 bits controlado por el registro SPBRG (99h). El bit 2 del registro TXSTA (BRGH) determina el valor del baud rate según la siguiente fórmula:

- Si BRGH = 0 (baja velocidad de transmisión):

$$Baud\ Rate = \frac{Foscilador}{16 \times (X + 1)}$$

- Si BRGH = 1 (alta velocidad de transmisión):

$$Baud\ Rate = \frac{Foscilador}{64 \times (X + 1)}$$

Siendo X, en ambos casos, el valor de 8 bits del registro SPBRG. El valor de éste registro se puede obtener a partir de la siguiente tabla, sabiendo que la transmisión se va a realizar a 9600 baudios, que el oscilador tiene una f = 4MHz y se quiere una alta velocidad de transmisión.

BAUD RATE (K)	Fosc = 4 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-
1.2	1.202	0.17	207
2.4	2.404	0.17	103
9.6	9.615	0.16	25
19.2	19.231	0.16	12
28.8	27.798	3.55	8
33.6	35.714	6.29	6
57.6	62.500	8.51	3
HIGH	0.977	-	255
LOW	250.000	-	0

Tabla 2: Valor del registro SPBRG para transmisión asíncrona.

Se obtiene que el valor del registro SPBRG = 25.

- Transmisor asíncrono:

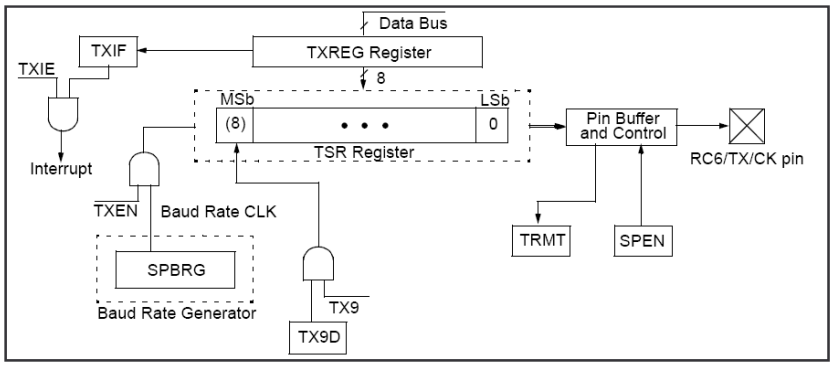


Ilustración 16: Diagrama de bloques de transmisión de datos (USART).

Para habilitar el módulo de transmisión hay que poner a nivel lógico '1' el bit 5 (TXEN) del registro TXSTA. La transmisión de los datos se realizaría por el pin 17 del micro (RC6/TX/CK).

Para transmitir un dato, deberá escribirse primero en el registro TXREG (19h). Este dato se transmite al registro TSR (registro que contiene la trama de datos a enviar). Cuando el TSR termina de enviar la trama (en cuanto transmite el bit de paro) vuelve a leer otro dato que contenga TXREG (si hay alguno). En cuanto el dato de TXREG es transferido al TSR el TXREG queda vacío esta condición es indicada mediante el bit TXIF (bit 4 del registro PIR1 (0Ch)), el cual se pone a nivel lógico '1'.

- Receptor asíncrono:

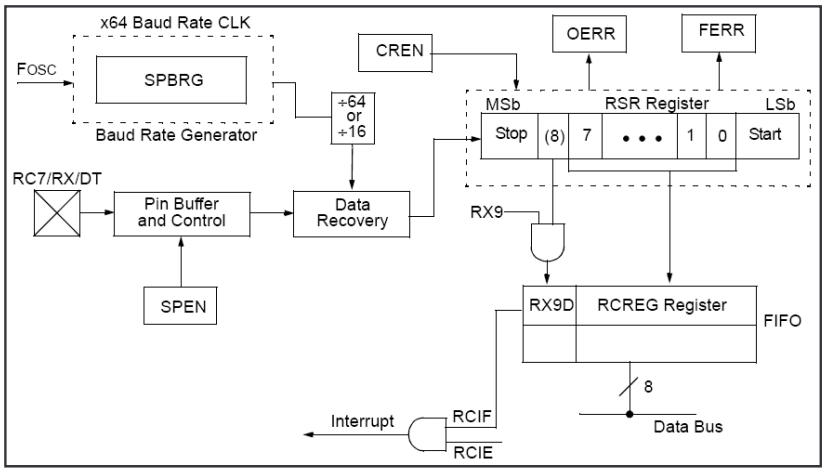


Ilustración 17: Diagrama de bloques de recepción de datos (USART).

La recepción se habilita poniendo a nivel lógico '1' el bit 4 (CREN) del registro RCSTA. El dato se recibe por el pin 18 del micro (RC7/RX/DT) y se almacena en el registro RSR. Cuando el último bit de la trama es recibido, el dato es transferido al registro RCREG (1Ah) (si éste está vacío, puede almacenar hasta dos datos) y al mismo tiempo el bit 5 del registro PIR1 (RCIF) se pone a nivel lógico '1'. Cuando se han leído los datos del registro RCREG este bit (RCIF) se pone a nivel lógico '0'.

2.2.2.4.4 Módulo Master Synchronous Serial Port (MSSP):

Según se establece en el estándar de comunicación I^2C, para la comunicación entre dispositivos, uno de ellos actúa como maestro (placa de control) y el resto actúan como esclavos (telémetros y brújula) cada uno de ellos con una dirección asociada por software.

Por consiguiente, el módulo MSSP del micro se debe configurar para operar en modo I^2C esclavo con dirección de 7 bits (hasta 128 dispositivos direccionables), poniendo a nivel lógico '1' el bit 5 (SSPEN) del registro SSPCON (14h) y el valor binario '0110' en los cuatro bits menos significativos del registro SSPCON. La velocidad de transmisión de los datos es de 400kHz. Los pines RC3 y RC4 del micro se deben configurar como entradas.

El diagrama de bloques del módulo es:

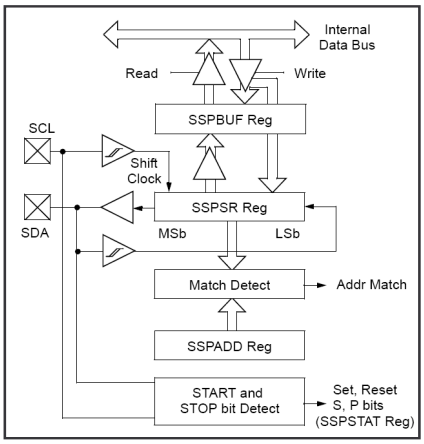


Ilustración 18: Diagrama de bloques módulo I^2C esclavo.

- Recepción de datos[1]: La dirección (de 7 bits) del medidor se carga en el registro SSPADD (93h). Cuando se ha recibido la condición de START, el siguiente byte (byte de dirección) es cargado en el registro SSPSR (no accesible), y se comparan los 7 bits más significativos del byte con la dirección del registro SSPADD. Si la dirección es la misma, se genera un bit de reconocimiento (ACK, generado por hardware) y se pone a nivel lógico '1' el bit asociado a la interrupción que se produce (SSPIF bit 3 del registro PIR1). El bit menos significativo del byte de dirección se carga en el registro SSPSTAT (94h) y si tiene nivel lógico '0' indica que el maestro va a enviar datos. Luego se leen los datos y se genera el bit (ACK) y se produce la interrupción por cada byte de datos recibido. Cada dato recibido se carga en el registro SSPBUF de donde es leído. El bit SSPIF se debe borrar (poner a nivel lógico '0') por software. Finalmente el maestro generará la condición de STOP.
- Transmisión de datos: La dirección (de 7 bits) del medidor se carga en el registro SSPADD (93h). Cuando se ha recibido la condición de START, el siguiente byte (byte de dirección) es cargado en el registro SSPSR, y se comparan los 7 bits más significativos del byte con la dirección del registro SSPADD. Si la dirección es la misma, se genera un bit de reconocimiento (ACK) y se pone a nivel lógico '1' el bit asociado a la interrupción que se produce (SSPIF bit 3 del

[1] Ver apartado 3.1 para conocer la configuración de las tramas de datos.

registro PIR1). El bit menos significativo se carga en el registro SSPSTAT (94h) y si tiene nivel lógico '1' indica que el maestro quiere recibir datos. Cada byte de datos a enviar se carga en el registro SSPBUF (13h) y posteriormente se pasa al registro SSPSR, se espera a que el maestro permita el envío de los datos y se envían. Se produce una interrupción por cada byte de datos enviado. El bit SSPIF se debe borrar (poner a nivel lógico '0') por software. Cada vez que el maestro recibe un dato genera un bit (ACK) para identificar que ha recibido el dato de forma correcta. Cuando el maestro reciba el último dato generará un bit ($\overline{ACK}$) y generará la condición de STOP.

2.2.2.5 Esquema de conexión de los terminales de los sensores a los puertos E/S del micro:

La distribución de los terminales de los ocho sensores (por placa) en los pines de los puertos A, B y C del micro se observa en la siguiente ilustración:

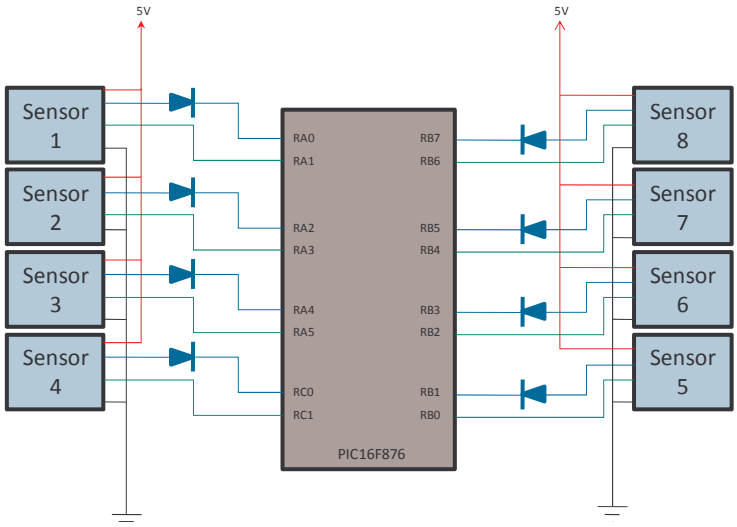


Ilustración 19: Conexiones sensores a los puertos del micro.

En el pin RA4 hay que colocar una resistencia de pull-up (100Ω) puesto que es una entrada a drenador abierto.

Como se ha explicado anteriormente y se puede ver en la Ilustración 19 y en la Ilustración 20, se conecta un diodo limitador entre la salida del micro y el terminal de entrada del sensor, para proteger a este último. Se utiliza un diodo de alta velocidad para evitar acumular excesivos retardos en la obtención de la medida. Se utilizan 8 diodos

por placa, uno por cada sensor. El tiempo de conmutación del diodo es de 4ns máximo. Se pueden ver más características de este diodo en el ANEXO A.

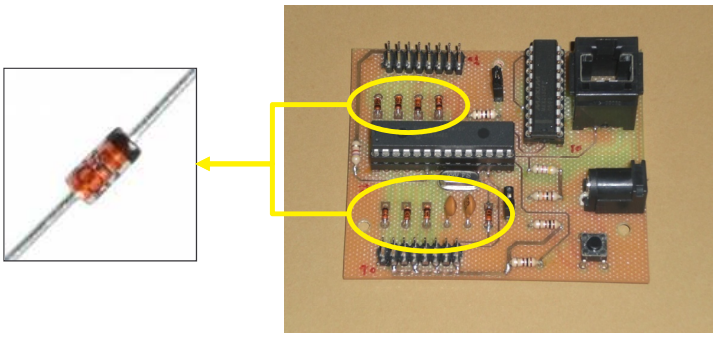


Ilustración 20: Diodo 1N4148.

2.2.3 Conectores

2.2.3.1 Conexión a los sensores:

Cada sensor tiene cuatro terminales, es decir, cada puerta, de la base del robot, que tiene ocho sensores, tendrá 32 terminales. Todos los terminales se reúnen en dos conectores hembra para cable plano.

Debido a esto, se integran en las placas dos conectores macho (8x2 pines) para cable plano (16 hilos), uno a cada lado de la placa, para conectar los sensores a la misma (4 sensores en cada conector). El esquema de conexiones sería:

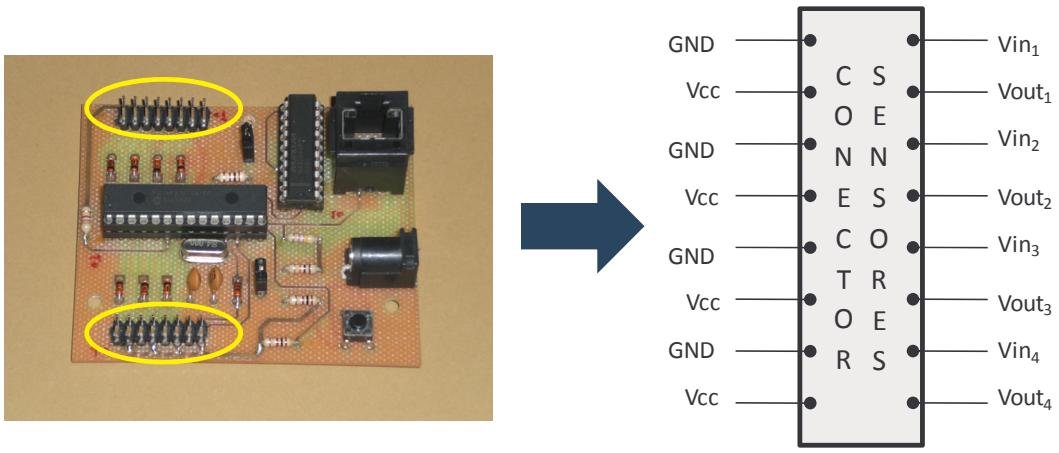


Ilustración 21: Conectores a sensores.

De esta forma la conexión entre la placa y los sensores puede ser directa (conectando el conector hembra de los sensores con el conector macho de la placa) o indirecta por medio de un cable plano de 16 hilos y los conectores correspondientes a cada lado del cable plano, tal y como se muestra en la siguiente ilustración:

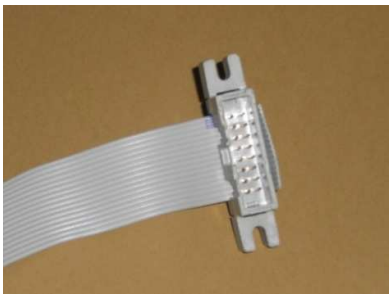


Ilustración 22: Conector y cable plano de 16 hilos.

2.2.3.2 Conexión al PC o a la placa de control:

Otro conector que se utiliza, para tener integrados todos los terminales de comunicaciones en un solo conector, es el RJ45. La distribución del patillaje del mismo es:

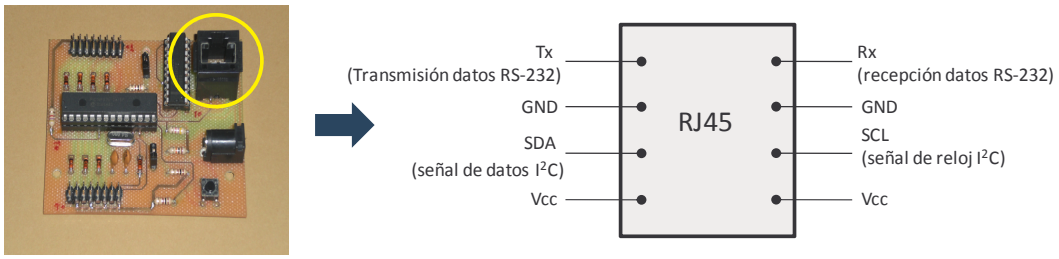


Ilustración 23: Conector medidor-placa de control o medidor-PC.

2.2.3.3 Alimentación:

Los medidores pueden estar alimentados de dos formas distintas. A través del conector RJ45, recibiendo la alimentación de la placa de control, o a través de un conector integrado en la placa del medidor. La redundancia de la alimentación permite que en caso de fallo de una de ellas, el sistema siga funcionando, si se alimenta desde la redundante. Para el correcto funcionamiento del medidor, éste debe ser alimentado a $+5V_{DC}$.

2.2.4 Selección de la denominación de puerta

Los medidores se sitúan uno en cada puerta de la base del robot. A efectos prácticos y para este proyecto las puertas se van a denominar puerta A y puerta B. Cada medidor puede ser configurado como puerta A o como puerta B, así, según se quiera, la puerta delantera puede ser la puerta A o B y lo mismo ocurriría con la puerta trasera. Esta selección de la denominación de puerta se establece mediante un jumper situado en el pin RC5 del micro en las dos placas.

- JUMPER CONECTADO (PIN_RC5 = 0V): puerta A.
- JUMPER NO CONECTADO (PIN_RC5 = 5V): puerta B.

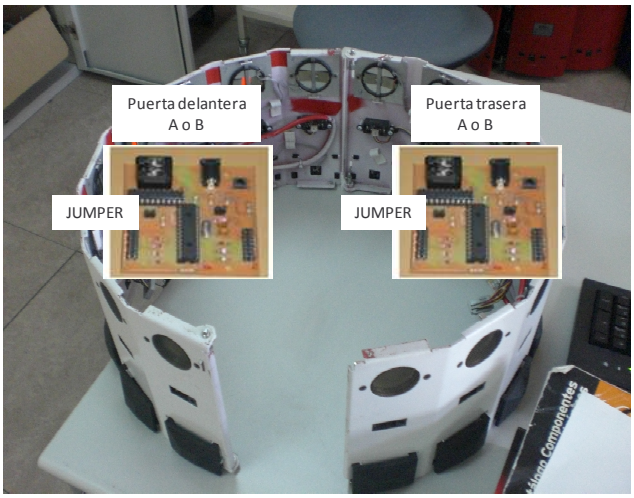


Ilustración 24: Puertas A o B de la base del robot.

2.3 Integración del telémetro por ultrasonidos

Este apartado se ha redactado teniendo en cuenta el proyecto titulado "Sistema sensorial para robots móviles basado en multiplexación de sensores sonar" [14].

En él se ha desarrollado un telémetro por ultrasonidos (Ilustración 25), capaz de medir distancias hasta 10 metros aproximadamente.

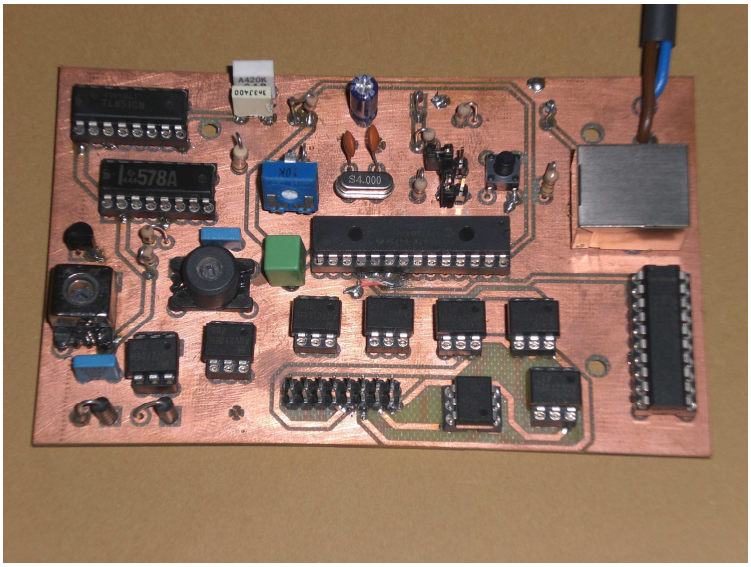


Ilustración 25: Telémetro por ultrasonidos.

Los sensores de ultrasonidos son de la serie 600 de Polaroid que entre sus características están: rango de medida de 15cm a 10.7m, la tensión de funcionamiento es igual a $400V_{DC}$, consumo de 15 mA en funcionamiento y de 3mA en reposo…

El principio de funcionamiento del telémetro por ultrasonidos se basa en emitir un tren de pulsos ultrasónicos con una frecuencia de 49.4KHz y esperar hasta que se recibe el rebote de la señal. Midiendo el tiempo entre la emisión y la recepción, se obtiene la distancia al objeto. Esta medición se calcula teniendo en cuenta la velocidad del sonido en el aire.

El sistema de adquisición de los datos de los sensores de ultrasonidos diseñado consta de tres partes:

- Circuito de excitación: donde se genera la señal ultrasónica y está formado por el integrado TL851 y un transformador que permite inducir una alta tensión a la entrada del sensor.
- Circuito de adquisición: encargado de recibir el eco (señal de retorno) y filtrar la señal para eliminar el ruido. Está formado por el integrado TL852 y otros elementos analógicos.
- Circuito de multiplexación: formado por ocho photoMOS de alto voltaje controlados externamente por un micro y permite que la señal ultrasónica pueda ser distribuida a los sensores de ultrasonidos.

El telémetro por ultrasonidos ha sido diseñado para permitir su comunicación serie según el protocolo I^2C y según el protocolo RS-232. Por lo tanto permitirá su comunicación con la placa de control y pruebas de funcionamiento por medio de un PC.

El dato que el telémetro transmite a la placa de control es el tiempo de envío y recepción de la señal, posteriormente y, tras la calibración, se puede obtener el valor de la distancia al objeto. En el apartado 4.3 se detalla con más detalle esta cuestión.

2.4 Integración de una brújula digital ("módulo CMPS03")

En este apartado se va a describir el módulo CMPS03 describiendo sus componentes, así como la manera de obtener la orientación del robot respecto al Norte magnético.

Las dimensiones del módulo son: 30x30mm.

El módulo CMPS03 tiene integrados dos sensores magnéticos (KMZ51[1] de Philips) que tienen la sensibilidad suficiente para detectar el campo magnético terrestre, actuando de esta forma como brújula digital. Los sensores se encuentran colocados en ángulo de 90º (coordenadas *X* e *Y*). Dispone también de un microcontrolador PIC18F2321[2], para registrar los valores proporcionados por los sensores.

La brújula digital es ideal para aplicaciones en donde se necesite visualizar o controlar la dirección, como, por ejemplo, un robot que vaya determinando y siguiendo el punto cardinal que se le ha asignado como destino.

2.4.1 Principio de funcionamiento

Por medio de los sensores se tienen dos coordenadas *X* e *Y*. Cada coordenada reporta la fuerza de la componente de campo magnético paralela a ésta. La coordenada *X* reporta $X * cos\theta$, y la coordenada *Y* reporta $X * sen\theta$. Para hallar θ (que es el ángulo de inclinación entre la parte frontal del módulo y el Norte en sentido horario) se usa la $\tan^{-1} -Y/X$ con lo cual se obtiene fácilmente el punto cardinal hacia el cual se dirige el robot, a través de una escala de 0º - 360º donde el cero representa al Norte magnético.

En las siguientes ilustraciones se puede observar la configuración del módulo y las conexiones de los diferentes pines:

[1] Más información: http://www.datasheetcatalog.com/datasheets_pdf/K/M/Z/5/KMZ51.shtml
[2] Más información: http://www.microchip.com

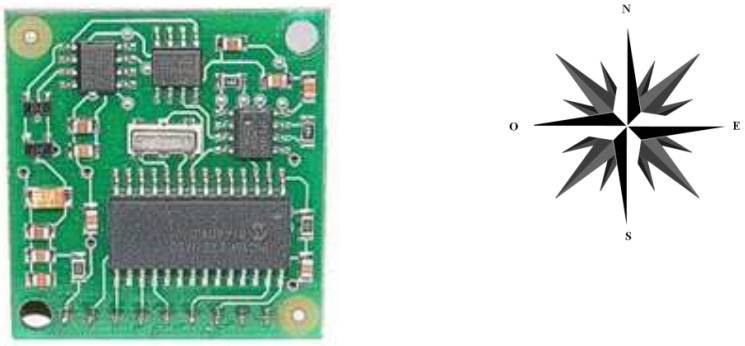


Ilustración 26: Orientación de la brújula digital (Módulo CMPS03).

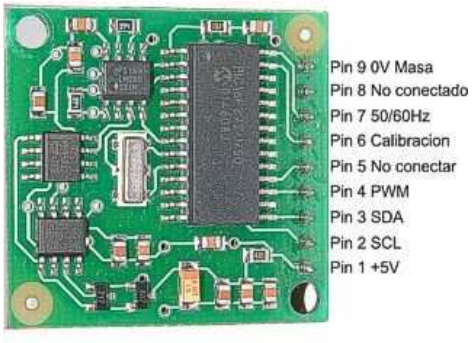


Ilustración 27: Patillaje del módulo CMPS03.

El punto cardinal determinado por la brújula se puede obtener de dos formas diferentes:

- Por medio del pin 4: salida en la que se obtiene una señal PWM, en la que el pulso positivo representa el ángulo de la brújula. El pulso varía en duración desde 1mS (0º) hasta 36,99 ms (359,9 º), o dicho de otra forma, el pulso es igual a 100 µS × ºC + 1ms de tara. La señal permanece a nivel lógico '0' durante 65 ms entre pulsos, por lo que el período de trabajo es de 65mS más la anchura del pulso. El pulso es generado por un contador de 16 bits del propio micro, con una resolución de 1 µS, aunque en la práctica no es recomendable hacer mediciones con una resolución de más de 0,1º (10µS). Para obtener la medida a través del pin 4, es necesario conectar a +5V mediante 2 resistencias de 47 KΩ, los pines 2 y 3 (SCL - SDA) del interfaz I^2C, ya que no se incluyen resistencias de pull-up en el circuito [7].

- Por medio de la comunicación con otros dispositivos: siguiendo el protocolo I^2C, método elegido en este proyecto. Los pines 2 y 3 (no tienen resistencias de pull-up[1] necesarias según el protocolo) permiten una lectura directa del valor en grados de la dirección [7]. En el apartado "Protocolo de comunicación entre dispositivos. Nivel físico" de este mismo capítulo se verá el hardware necesario para la comunicación I^2C y en el capítulo 3 "Implementación del software" se explicará cómo se define el protocolo en cuanto a la transmisión y recepción de datos.

El pin 7 se utiliza para seleccionar entre 50Hz (PIN7 = 0V) ó 60Hz (PIN7 = 5V). Esto se realiza debido a una desviación errónea de 1,5º causada por el campo generado por la red eléctrica. Sincronizando la conversión con la frecuencia en hertzios de la red, se consigue disminuir el error a tan solo 0,2º. El pin 7 tiene una resistencia interna de pull-up, por lo que si se deja sin conectar, funcionará a 60Hz [8].

El micro del módulo tiene un total de 16 bytes de registros, algunos de los cuales forman registros de 2 bytes tal y como puede verse en la siguiente tabla:

REGISTRO	CONTENIDO
0	Nº Revisión Software
1	Dirección (0º - 360º) en 1 byte (0-255).
2,3	Dirección (0º - 359,9º) en 2 bytes (0-3599).
4,5	Test interno señal diferencial sensor 1
6,7	Test interno señal diferencial sensor 2
8,9	Test interno, valor calibración sensor 1
10,11	Test interno, valor calibración sensor 2
12,13,14	Sin usar, devuelven el valor 0
15	Comando de calibración

Tabla 3: Registros del micro del módulo CMPS03.

Para leer la dirección determinada por lo sensores magnéticos, se realizará una lectura del registro 2, de 16 bits.

[1] Resistencia conectada entre la señal y Vcc, su misión es elevar la tensión de salida de un dispositivo lógico.

2.5 Descripción hardware de la placa de control sensorial

En este apartado se van a describir todos los componentes que se han utilizado en el diseño y desarrollo de la placa de control del sistema sensorial de la base del robot. Esta placa tiene dos propósitos: actuar como interfaz entre el sistema sensorial (telémetros y brújula) y el PC del robot y controlar las secuencias de disparo de los diferentes sensores, almacenando los resultados obtenidos por los mismos.

Se ha diseñado una placa, que irá integrada también en la base del robot. Las dimensiones de la placa son: 87x98mm. En el ANEXO B se puede ver el esquemático y el layout de la placa, diseñados ambos con el programa ORCAD v10.0.

En la ilustración, se puede observar la placa de control:

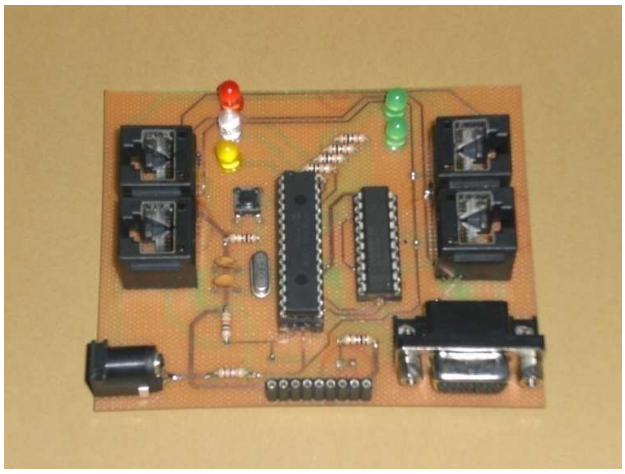


Ilustración 28: Placa de control.

2.5.1 Microcontrolador para la integración de los telémetros y la brújula: PIC18F2550

Para la placa de control se utiliza el microcontrolador PIC18F2550. Además de tener mayor capacidad de memoria que el PIC16F876 (permite realizar programas con mayor número de instrucciones) tiene la posibilidad de comunicación USB con otros dispositivos. Esto se puede utilizar en trabajos futuros, si se quiere cambiar de tipo de comunicación y se quiere implementar el protocolo de comunicación USB entre la placa de control y el PC del robot.

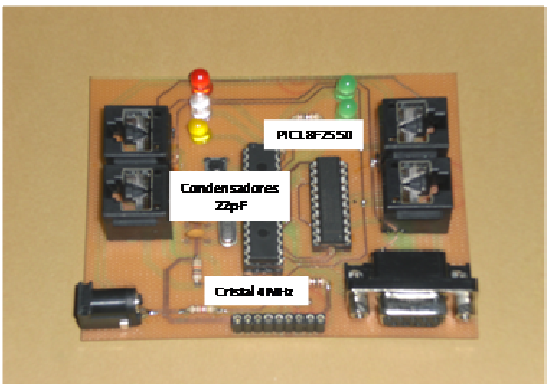


Ilustración 29: PIC18F2550 y circuito oscilador.

Como se puede observar, tiene 28 pines y encapsulado PDIP. Los siguientes apartados se han escrito teniendo en cuenta el manual del PIC [10].

Entre sus características principales cabe destacar:

- Micro de 8 bits (bus de datos).
- Memoria CMOS FLASH de 32Kx14bits (corresponde a la F del nombre).
- Memoria SRAM de 2048 bytes.
- Memoria EEPROM de 256 bytes.
- 24 entradas/salidas digitales distribuidas en 4 puertos (A, B, C y E) y 12 entradas analógicas disponibles en el puerto A y B.
- 4 contadores/temporizadores, tres de 8 bits y uno de 16 bits.
- Convertidor A/D → de 10 bits/10 canales.
- 2 CCP → Módulos Capture/Compare/PWM.
- Un puerto serie síncrono → Puede ser configurado como Serial Peripheral Interface (SPI) o como Inter-Integrated Circuit (I^2C).
- Un Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART).
- Un módulo Universal Serial Bus (USB v2.0).
- 19 Interrupciones.
- 75 Instrucciones.
- Voltaje (Vdd): 2 - 5,5V.

El oscilador y el circuito de reset utilizados para este micro, son los mismos que para el PIC16F876 utilizado en el medidor por infrarrojos[1].

2.5.1.1 Memoria de datos SRAM:

La memoria de datos consta de dos áreas mezcladas y destinadas a distintas funciones:

- Registros de Función Especial (SFR).
- Registros de Propósito General (GPR).

Los SFR son posiciones de memoria de 12 bits asociadas específicamente a los diferentes periféricos y funciones de configuración del PIC y tienen un nombre específico asociado con su función. Mientras que los GPR son posiciones de memoria RAM de uso general.

2.5.1.2 Periféricos integrados en el micro y utilizados en este proyecto:

2.5.1.2.1 Puertos I/O:

Los puertos del micro son totalmente programables, es decir, sus líneas pueden ser configuradas para trabajar como entradas o como salidas a elección del programador. Los puertos que se utilizan son:

- Puerto B: tiene 8 líneas bidireccionales. Los registros asociados al puerto son PORTB (F81h), TRISB (F93h), LATB (F8Ah) e INTCON2 (FF1h).

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP

Los registros PORTB y LATB permiten leer/escribir en todos los pines del puerto. El registro TRISB permite configurar la línea física homónima como entrada (nivel lógico ‘1’) o como salida (nivel lógico ‘0’). El bit más significativo del registro

[1] Ver apartados 2.2.2.1 y 2.2.2.2 de este capítulo.

INTCON2 ($\overline{RBPU}$) permite acoplar/desacoplar las resitencias de pull-up asociadas al puerto B.

- Puerto C: tiene 7 líneas bidireccionales. Los registros asociados al puerto son PORTC (F82h), TRISC (F94h) y LATC (F8Bh).

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTC	RC7	RC6	RC5	RC4	—	RC2	RC1	RC0
LATC	LATC7	LATC6	—	—	—	LATC2	LATC1	LATC0
TRISC	TRISC7	TRISC6	—	—	—	TRISC2	TRISC1	TRISC0

Los registros PORTC y LATC permiten leer/escribir en todos los pines del puerto. El registro TRISC permite configurar la línea física homónima como entrada (nivel lógico '1') o como salida (nivel lógico '0').

2.5.1.2.2 El puerto serie EUSART:

La EUSART puede ser configurada para operar como modo asíncrono full-dúplex o modo síncrono half-dúplex.

Este módulo presenta ciertas mejoras respecto al módulo homónimo del PIC16F876, como, por ejemplo, una detección y calibración automática del baud rate. Las operaciones de transmisión y recepción de datos son controladas por tres registros: TXSTA (FACh), RCSTA (FABh) y BAUDCON (FB8h). Éstas se establecen de la misma forma que para el PIC16F876 (ver apartado 2.2.2.4.3).

2.5.1.2.3 Módulo Master Synchronous Serial Port (MSSP):

El módulo MSSP del micro se debe configurar para operar en modo I^2C maestro[1], poniendo a nivel lógico '1' el bit 5 (SSPEN) del registro SSPCON1 (FC6h) y el valor binario '1000' en los cuatro bits menos significativos del registro SSPCON. La velocidad de transmisión de los datos es de 400kHz. Los pines RC3 y RC4 del micro se deben configurar como entradas.

[1] Este módulo acepta la opción de multi-maestro, pero no se utiliza para este proyecto

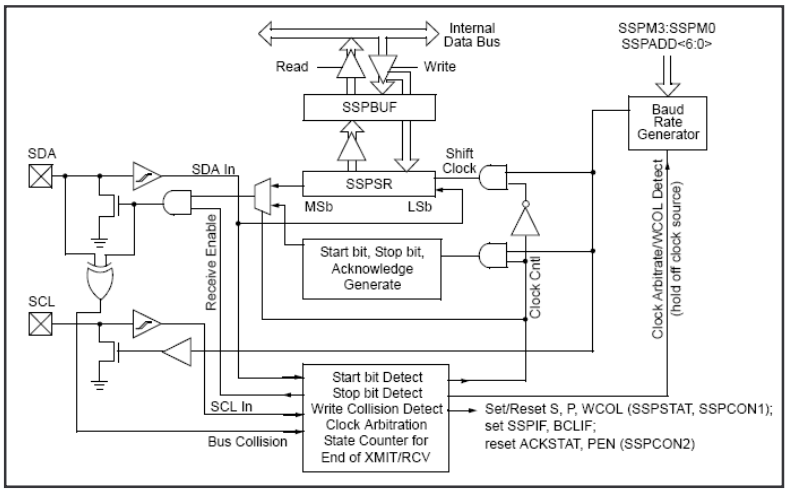


Ilustración 30: Diagrama de bloques módulo I^2C maestro.

Las líneas SCL y SDA son manejadas por el hardware del MSSP. El módulo genera los pulsos de reloj en la línea SCL y las condiciones de START, RE-START y STOP. La frecuencia del reloj se determina con el generador de baud rate, ésta puede ser de 100kHz, 400kHz y 1MHz (en este caso la segunda opción).

- Transmisión de datos: Después de generar la condición de START (poniendo a nivel lógico '1' el bit SEN del registro SSPCON2 (FC5h)), la dirección del esclavo (7 bits) se carga en el registro SSPBUF (FC9h) y el bit $R/\overline{W}$ se pone a nivel lógico '0'. Todo el byte es enviado. Cuando el ACK del esclavo es recibido se carga el registro SSPBUF con el byte de datos y se envía, se espera a recibir la confirmación del esclavo y se siguen enviando datos o se genera la condición de STOP (poniendo a nivel lógico '1' el bit PEN del registro SSPCON2).
- Recepción de datos: esta opción es activada poniendo a nivel lógico '1' el bit RCEN del registro SSPCON2. Después de generar la condición de START (poniendo a nivel lógico '1' el bit SEN del registro SSPCON2), la dirección del esclavo (7 bits) se carga en el registro SSPBUF (FC9h) y el bit $R/\overline{W}$ se pone a nivel lógico '1'. Todo el byte es enviado. Cuando el ACK del esclavo es recibido se carga el registro SSPSR con el dato recibido y pasa al registro SSPBUF de donde es leído. Por cada byte recibido se genera un bit ACK.

Cuando se han recibido todos los datos se genera un bit $\overline{ACK}$ y la condición de STOP (poniendo a nivel lógico '1' el bit PEN del registro SSPCON2).

2.5.2 Conectores

2.5.2.1 Conexión de la placa de control con telémetros sonar e IR:

El conector para permitir la comunicación entre la placa de control y los telémetros es el RJ45. Como hay cuatro telémetros que se conectan a la placa de control, se utilizan 4 conectores RJ45. La distribución del patillaje de estos conectores sería:

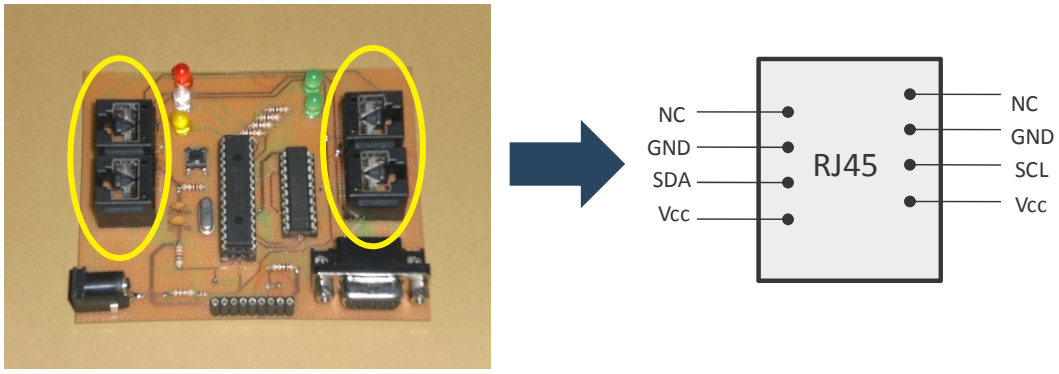


Ilustración 31: Conectores de la placa de control con telémetros.

Los pines 1 y 2 del conector se dejan libres, sin conectar (NC significa no conectado).

2.5.2.2 Conexión de la placa de control con la brújula:

El módulo CMPS03 tiene sus terminales agrupados en una tira de 9 pines macho, de tal forma, que para conectar éste, a la placa de control, se utiliza una tira de 9 pines hembra torneados. En la ilustración se señala el lugar donde va conectada la brújula dentro de la placa de control.

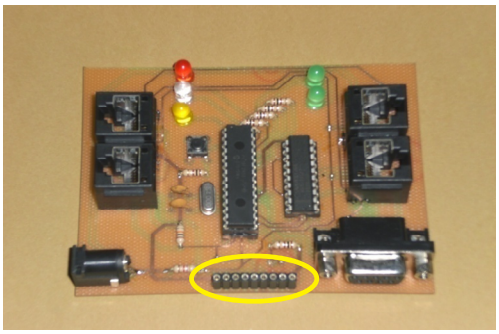


Ilustración 32: Conector placa control-brújula.

Si al integrar todo el sistema en la estructura del robot, hay problemas con los campos magnéticos que no permiten realizar una lectura correcta de la brújula, se podría adaptar este conector o encontrar un sitio en el robot con baja influencia de campos magnéticos.

2.5.2.3 Conexión de la placa de control con el PC del robot:

Como la comunicación entre la placa de control y el PC se establece según el estándar RS-232, se utiliza como conector un DB9 hembra (del que sólo se utilizan tres de sus terminales). La colocación del mismo en la placa, así como el esquema de sus terminales se puede observar en la siguiente ilustración:

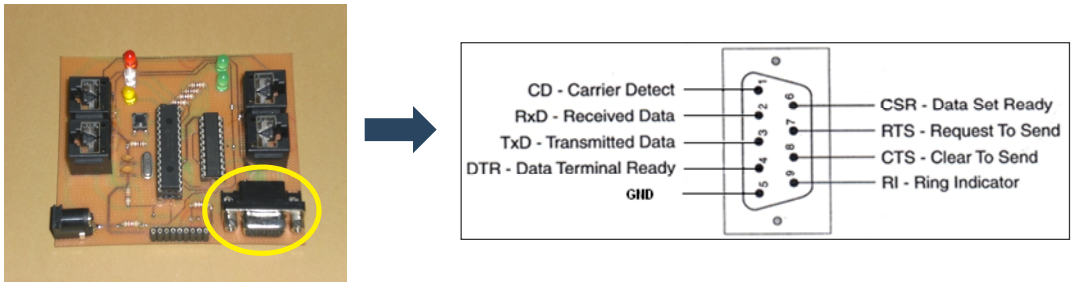


Ilustración 33: Terminales conector DB9.

En el apartado 2.6.2 se describe con mayor detalle el estándar de comunicación y las líneas que se utilizan del mismo.

2.5.2.4 Alimentación:

La placa de control puede estar alimentada de dos formas distintas. A través del conector RJ45, recibiendo la alimentación de cualquiera de los medidores, o a través de un conector integrado en la misma. La redundancia de la alimentación permite que en caso de fallo de una de ellas, el sistema siga funcionando, si se alimenta desde la redundante.

Para el correcto funcionamiento de la placa de control, ésta debe ser alimentada a $+5V_{DC}$.

2.5.3 Sistema de visualización del funcionamiento del sistema

Para detectar fallos producidos durante el funcionamiento del sistema, se ha incluido en la placa de control un sistema de LEDs. Si se observa el conexionado en la siguiente ilustración, se puede deducir que los LEDs son activos (emiten luz) cuando la salida correspondiente del micro se pone a nivel lógico '1' (activos a nivel alto). Todos los LEDs se han conectado al puerto B y no será necesario activar las resistencias de pull-up del mismo, porque todos los pines se declararán como salidas.

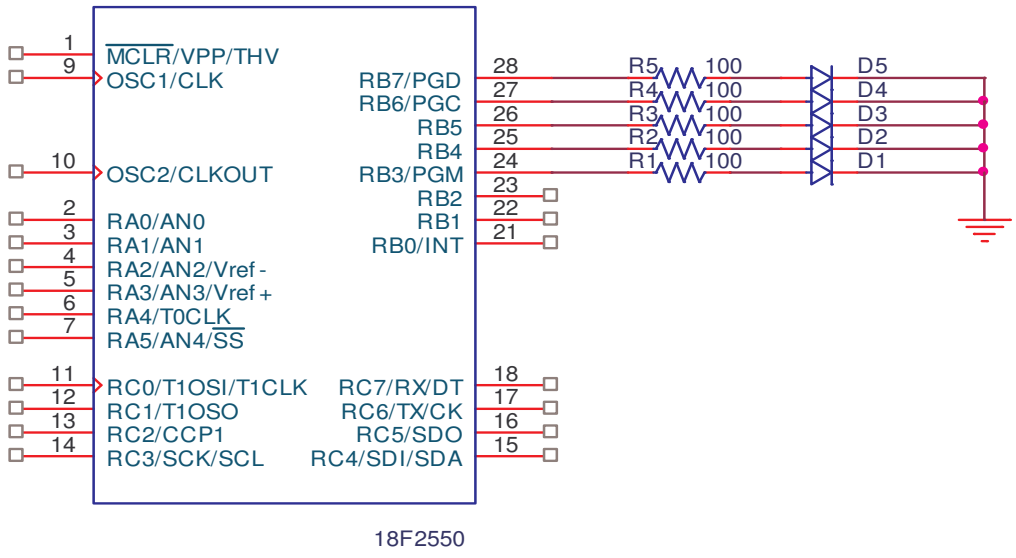


Ilustración 34: Conexión de los LEDs al PIC18F2550.

Al emitir luz indican:

- LED rojo: se está realizando una operación en el telémetro por ultrasonidos.
- LED azul: se está realizando una operación en el telémetro por infrarrojos.
- LED amarillo: se está realizando una operación en la brújula digital.
- LED verde superior: el medidor correspondiente está situado en la puerta delantera.
- LED verde inferior: el medidor correspondiente está situado en la puerta trasera.

Esto permite detectar con facilidad cualquier fallo que se produzca en el sistema, identificando con claridad y rapidez el elemento que falla.

Cuando se produce algún fallo en cualquiera de los elementos que componen el sistema, se puede desconectar dicho elemento de la placa de control. A continuación se resetea la misma y el sistema sigue funcionando (salvo excepciones[1]) avisando que está desconectado el elemento en concreto. Una vez reparado el fallo se podría volver a conectar el elemento y volviendo a poner en marcha todo el sistema se continuaría con el funcionamiento normal del sistema.

Todo este desarrollo permite tener un sistema robusto y modular que es uno de los objetivos marcados para la realización de este proyecto.

[1] Ver apartado 3.2.2 para más información

2.6 Protocolo de comunicación entre dispositivos. Nivel físico

Como se ha mencionado anteriormente, los telémetros se pueden comunicar según dos estándares diferentes, en función del dispositivo al que se conecten (PC o placa de control). La selección de uno u otro estándar se realiza mediante el jumper situado en el pin RC2 del microcontrolador:

- JUMPER CONECTADO: comunicación I^2C (PIN RC2 = 0V).
- JUMPER NO CONECTADO: comunicación RS-232 (PIN RC2 = 5V).

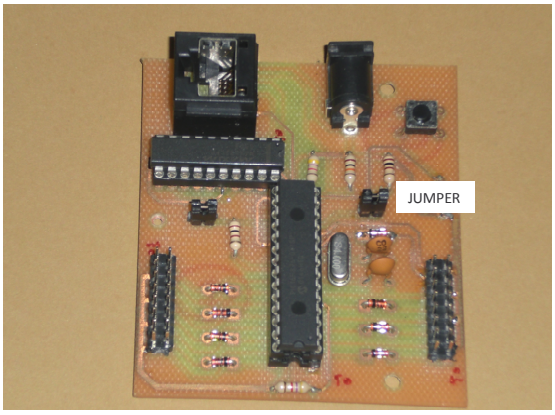


Ilustración 35: Selección del tipo de comunicación.

En este apartado se van a describir los estándares de comunicación utilizados en este proyecto, desde el punto de vista físico, es decir, el hardware necesario para poder establecer la comunicación.

2.6.1 Estándar de comunicación I^2C

Comunicación serie síncrona, es decir, misma señal de reloj para todos los dispositivos. La comunicación es half-duplex, usando niveles TTL (0V, +5V). El control del flujo de los datos se realiza por software.

Según las especificaciones del mismo, se requieren dos líneas, una línea de datos (SDA) y una línea que actúa como reloj (SCL), donde se genera el sincronismo de la

comunicación. Se requieren unas resistencias de pull-up en ambas líneas (no incluidas en los pines de ambos micros (PIC16F876 y PIC18F2550)). Se coloca una resistencia de 4,7KΩ en la línea SDA y una resistencia de 10KΩ en la línea SCL. En cuanto a características eléctricas, la tensión máxima para la que se asigna el valor lógico '0' es 1,5V y la tensión mínima para la que se asigna el valor lógico '1' es 3V.

Se utiliza para comunicar (transmitir/recibir datos) la placa de control con los medidores (sonar e infrarrojos) y la brújula digital.

2.6.2 Estándar de comunicación RS-232

Comunicación serie asíncrona, es decir, cada dispositivo tiene su propia señal de reloj. Se utiliza para comunicar dos dispositivos entre sí. La sincronización se consigue mediante bits de start y parada. La comunicación puede ser simplex[1], half-duplex (utilizada en el proyecto) o full-duplex, usando niveles de tensión bipolar (-12V, +12V). El estándar puede ser de 9 (más utilizado) o de 25 líneas. Como conectores se puede utilizar un DB9 o un DB25.

La conexión entre dispositivos será null-modem no-handshake[2], es decir, de las 9 líneas sólo se utilizan 3: una para transmitir datos (Tx), otra para recibir datos (Rx) y otra que tiene que ser común a ambos dispositivos que es tierra (GND). El esquema de conexión sería:

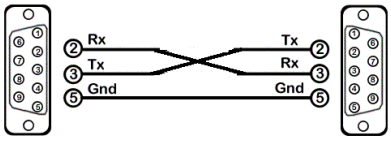


Ilustración 36: Conexión null-modem.

Esta comunicación se establece, de manera principal, para transferir datos (envío/recepción) entre la placa de control y el PC de *Maggie*. De forma menos importante, también se establece este estándar de comunicación para realizar pruebas de funcionamiento de los medidores en un PC.

[1] Los datos se envían en una única dirección.
[2] No hay control de flujo por hardware.

Según las especificaciones del estándar RS-232 se producen -12V para un '0' lógico y +12V para un '1' lógico. La salida del micro es 0V-5V, por lo que se necesita un adaptador de tensiones, se utiliza en este proyecto el MAX233.

2.6.2.1 Adaptador de tensiones:

El MAX233 es un adaptador de tensiones, que adapta los niveles de tensión de salida del micro (TTL/CMOS → 0V-5V) a los niveles de tensión del puerto COM (RS-232 → ±12V). El esquema de conexionado y diagrama de pines se muestran a continuación:

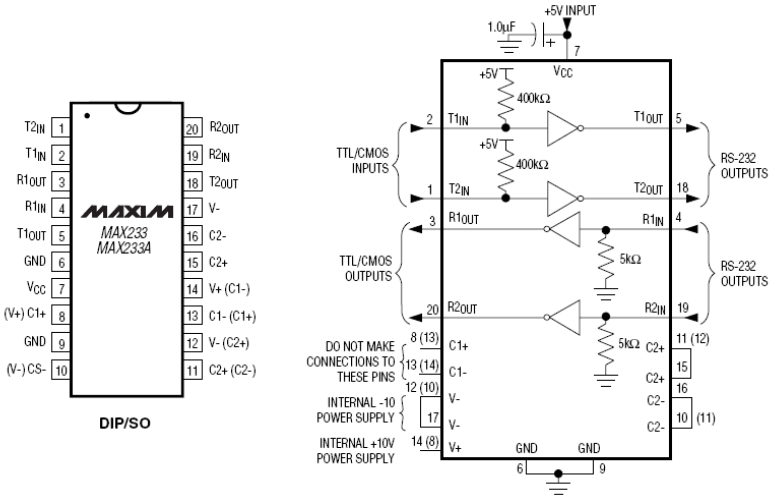


Ilustración 37: MAX233.

2.6.2.2 Adaptador RJ45-DB9:

Para poder establecer la comunicación entre el medidor y el puerto COM del PC se requiere un conector o cable adaptador de RJ45 (8 terminales) a DB9 (9 terminales), como el que se muestra en la ilustración:

Ilustración 38: Adaptador RJ45-DB9.

2.6.2.3 Adaptador Serie DB9 - USB 2.0:

Este adaptador va a permitir conectar la placa de control, que es un dispositivo serie, con el PC del robot a través de un puerto USB 2.0.

Ilustración 39: Adaptador serie-USB.

CAPÍTULO 3.- IMPLEMENTACIÓN DEL SOFTWARE

En este capítulo se describe el software que se ha tenido que desarrollar para obtener los objetivos marcados en el proyecto: obtener una visión de los objetos situados alrededor del robot para que éste pueda obtener información del entorno y desplazarse por el mismo con autonomía. En primer lugar se describen de forma general los programas utilizados para cada dispositivo (telémetro por infrarrojos y placa de control), así como las funciones que los componen. Al final del capítulo se describe el driver desarrollado para que la placa de control pueda comunicarse con el PC del robot, así como un programa para probar el funcionamiento del sistema.

El desarrollo del software de todo el sistema se ha realizado en lenguaje de programación de alto nivel, en concreto, en lenguaje C. Se pensó la posibilidad de realizarlo en lenguaje de bajo nivel o ensamblador (para que los programas ocuparan un

mínimo espacio de memoria en el micro), pero como se eligió un micro con memoria suficiente se prefirió descartar esta última opción.

Al elegir el lenguaje C, se necesitaba un compilador de C para microcontroladores PIC. Se encontraron dos posibilidades: HI-TECH PICC Compiler y CCS C Compiler. Una vez desarrollados varios programas de prueba y viendo las necesidades que exige el proyecto, se decidió utilizar el compilador de C de CCS, ya que se incluyen muchas funciones en librerías que permiten un fácil manejo de las comunicaciones. Como simulador se ha utilizado el "MPLAB IDE" desarrollado por Microchip Technology Inc. Integrando un plugin de CCS, en este simulador, se pueden utilizar ambas herramientas simultáneamente. Al compilar los programas se obtiene el archivo .hex que es el que se graba en la EEPROM del micro. Como programador se ha utilizado el MPLAB ICD2 comercializado, también, por Microchip Technology Inc.

3.1 Protocolos de comunicación. Tramas de datos

3.1.1 I^2C

Según establece el protocolo, los datos se transfieren en paquetes de 8 bits, en forma bidireccional y a tres velocidades posibles: normal (100kbits/s), rápida (400kbits/s) y alta velocidad (3,4Mbits/s). El número máximo de bytes que pueden ser enviados no está restringido, siendo el esclavo quien fija esta cantidad de acuerdo a sus características. El envío de bytes siempre se efectúa desde el bit MSB hasta el bit LSB.

Como ya se explicó en el capítulo anterior, la comunicación está formada por un dispositivo maestro (placa de control) y varios dispositivos esclavos (medidores y brújula digital) con una dirección única asignada a cada dispositivo esclavo. La dirección del esclavo puede ser de 7 o de 10 bits. El número máximo de dispositivos que se pueden conectar al bus está limitado a un máximo de 400pF [11], 128 ó 1024 dispositivos, para direcciones de 7 bits o 10 bits respectivamente.

Los datos se transmiten por la línea SDA y la línea SCL es la señal de reloj. La generación de los pulsos de reloj es siempre responsabilidad del maestro, al mismo tiempo es el encargado de empezar y terminar todas las transferencias de datos.

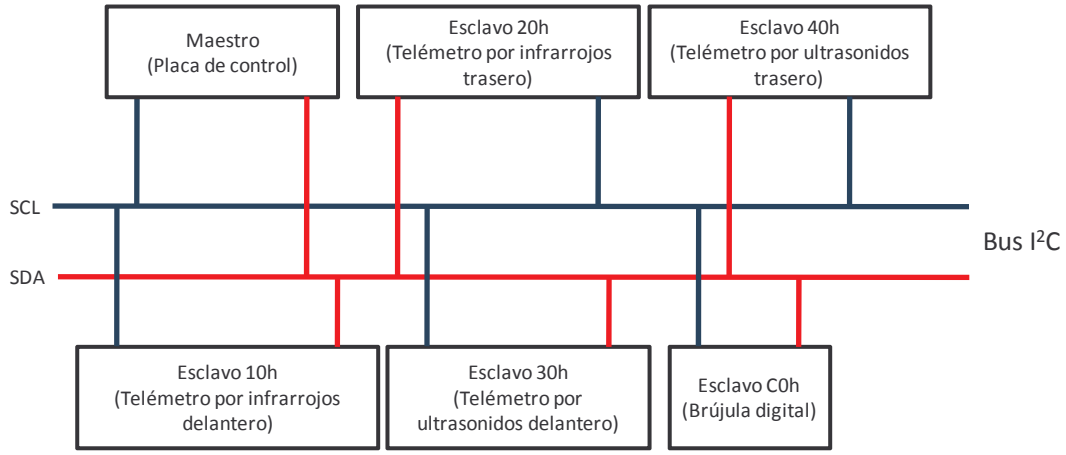


Ilustración 40: Distribución dispositivos en el bus I2C.

En la Ilustración 40 se puede observar la distribución de todos los dispositivos a lo largo del bus y que todos los dispositivos esclavos tienen una dirección asignada escrita en hexadecimal.

3.1.1.1 Maestro envía datos al esclavo:

El maestro actúa como transmisor de los datos y el esclavo correspondiente como receptor de los mismos. El sentido de flujo de los datos es del maestro hacia el esclavo y no cambia.

START Dirección esclavo + 0 ACK DATO ACK DATO ACK STOP

Una vez que el maestro ha generado la condición de START, el primer byte enviado es el byte de direccionamiento. Los siete primeros bits del byte representan la dirección del dispositivo esclavo con el que el maestro se quiere comunicar, el octavo bit ($R/\overline{W}$) representa la operación que quiere realizar el maestro (en este caso $R/\overline{W} = 0$ → el maestro envía datos al esclavo). Cuando este byte es enviado todos los dispositivos esclavos lo leen y lo comparan con su dirección de identificación y, si son iguales, el esclavo se considera direccionado como esclavo-receptor.

Después de haber recibido el bit de reconocimiento (ACK) por parte del esclavo, se envían los datos, recibiendo un bit de reconocimiento por cada dato enviado.

Finalmente el maestro genera la condición de STOP.

3.1.1.2 Maestro recibe datos del esclavo:

El maestro va a recibir datos del esclavo inmediatamente después de enviar el byte de direccionamiento. Una vez recibido el bit de reconocimiento por parte del esclavo, el maestro se configura como receptor y el esclavo como transmisor.

START | Dirección esclavo + 1 | ACK | DATO | ACK | DATO | $\overline{ACK}$ | STOP

Una vez que el maestro ha generado la condición de START, el primer byte enviado es el byte de direccionamiento. Los siete primeros bits del byte representan la dirección del dispositivo esclavo con el que el maestro se quiere comunicar, el octavo bit ($R/\overline{W}$) representa la operación que quiere realizar el maestro (en este caso $R/\overline{W} = 1$ → el maestro quiere recibir datos del esclavo). Cuando este byte es enviado todos los dispositivos esclavos lo leen y lo comparan con su dirección de identificación y, si son iguales, el esclavo se considera direccionado como esclavo-transmisor.

Cuando el esclavo empiece a transmitir los datos, es el maestro el que tiene que generar el bit de reconocimiento por cada byte recibido. Para detener la transferencia de datos, el maestro genera un bit ($\overline{ACK}$) y la condición de STOP.

3.1.1.3 Maestro envía/recibe datos del esclavo:

Las dos operaciones mencionadas anteriormente se pueden realizar en una única. En esta ocasión va existir un flujo de datos bidireccional, de esta forma, el dispositivo que en principio se comporta como transmisor después se comporta como receptor y viceversa. Esto implica que el maestro tiene que enviar dos veces la condición de START y en el byte de direccionamiento el bit $R/\overline{W}$ una vez será igual a cero y la otra será igual a uno.

START | Dirección $+ R/\overline{W}$ | ACK | DATOS | $ACK/\overline{ACK}$ | START | Dirección $+ R/\overline{W}$ | ACK | DATOS | $ACK/\overline{ACK}$ | STOP

3.1.2 RS-232

La comunicación se establece entre dos dispositivos, uno de ellos actúa como transmisor de los datos y el otro como receptor de los mismos. La conexión entre ellos va a ser null-modem y el control de flujo de los datos se establece vía software. La transmisión se inicia con un bit de comienzo, que sirve para sincronizar la comunicación, luego se envían los datos (comenzando por el LSB) y finalmente el bit de parada. Se pueden utilizar bits de paridad para determinar que la transmisión y recepción de los datos ha sido correcta.

Antes de establecer la comunicación hay que determinar una serie de parámetros comunes a ambos dispositivos: los baudios a los que se transmite, el número de bits de datos, tipo de paridad (si la hay) y número de bits de parada (1, 1 ½ ó 2).

Para este proyecto se establece una comunicación a 9600 baudios y 8N1, es decir, 8 bits de datos, sin paridad y un bit de parada.

3.2 Programas

Se puede consultar el ANEXO C para ver el código completo de los programas. En este apartado se van a explicar los programas desarrollados para la placa de control y para el telémetro por infrarrojos, mostrando, en primer lugar, el diagrama de flujo y, posteriormente, una breve explicación de los mismos.

3.2.1 Telémetro por infrarrojos

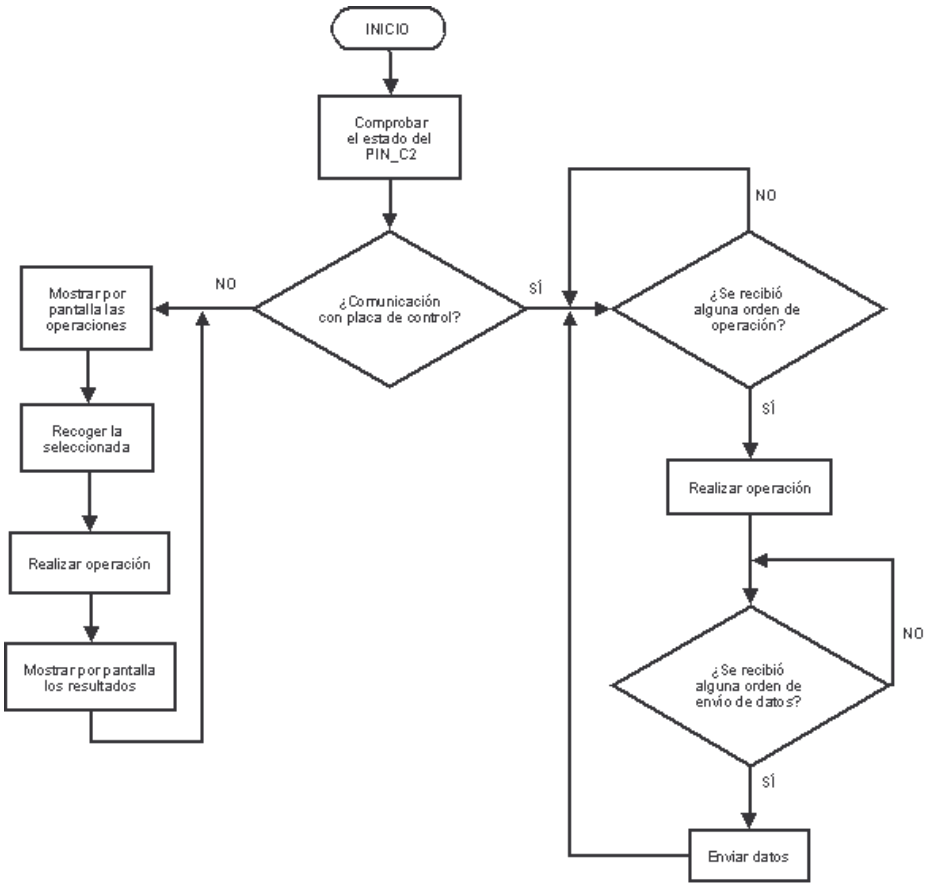


Ilustración 41: Diagrama de flujo programa telémetro por infrarrojos.

Las únicas diferencias entre el programa del medidor por infrarrojos delantero o frontal respecto al programa del medidor por infrarrojos trasero es la dirección de esclavo asignada y el número del sensor sobre el que se realiza la operación. Al medidor delantero se le asigna la dirección hexadecimal 10h y dispara los sensores del 1-8 y al medidor trasero se le asigna la dirección hexadecimal 20h y dispara los sensores del 9-16.

En el programa se definen, en primer lugar, todas las opciones de configuración, como el reset, la frecuencia de reloj, los registros del PIC, los pines del puerto A como E/S digitales, así como los parámetros de los módulos USART (velocidad de transmisión, pin Tx, pin Rx) e I^2C (dirección esclavo, velocidad de transmisión normal,

pin SCL, pin SDA) y se habilita la interrupción interna provocada por cada byte transmitido/recibido mediante I^2C.

Se comprueba si el jumper colocado en el pin RC2 del PIC16F876 está conectado o no, para determinar si la comunicación se establece con un PC para realizar pruebas de funcionamiento o con la placa de control (funcionamiento normal del sistema).

Si la comunicación se establece con la placa de control, el medidor se comporta como un dispositivo esclavo dentro del bus I^2C y tiene que esperar hasta que la placa de control (maestro) quiera que el medidor realice alguna operación. Una vez recibida la orden de operación, se realiza y se obtienen los valores de los sensores (DEC[1]), luego se espera la orden por parte del maestro para enviar estos valores obtenidos. Una vez transmitidos, se permanece en espera hasta que el maestro vuelva a direccionar al medidor.

Si la comunicación se establece con el PC (pruebas de funcionamiento del telémetro), se muestran por pantalla las operaciones que se pueden realizar en el medidor. Se lee la operación elegida por el usuario y se realiza, se obtienen los valores de los sensores (DEC), se convierte el valor mediante la fórmula del apartado 2.2.1.3 y se muestra por pantalla las distancias a los objetos (L^2). Se permanece en espera hasta que el usuario indique otra operación.

3.2.1.1 Recepción/envío de datos de/a la placa de control:

La placa de control va a enviar al medidor la operación que debe realizar y éste le va a enviar los resultados de la misma.

Por cada byte recibido desde la placa de control se genera una interrupción, lo mismo ocurre cada vez que se envía un byte a la placa de control.

[1] Valor de 8 bits (0-255) obtenido por el sensor de infrarrojos
[2] Distancia en cm al objeto.

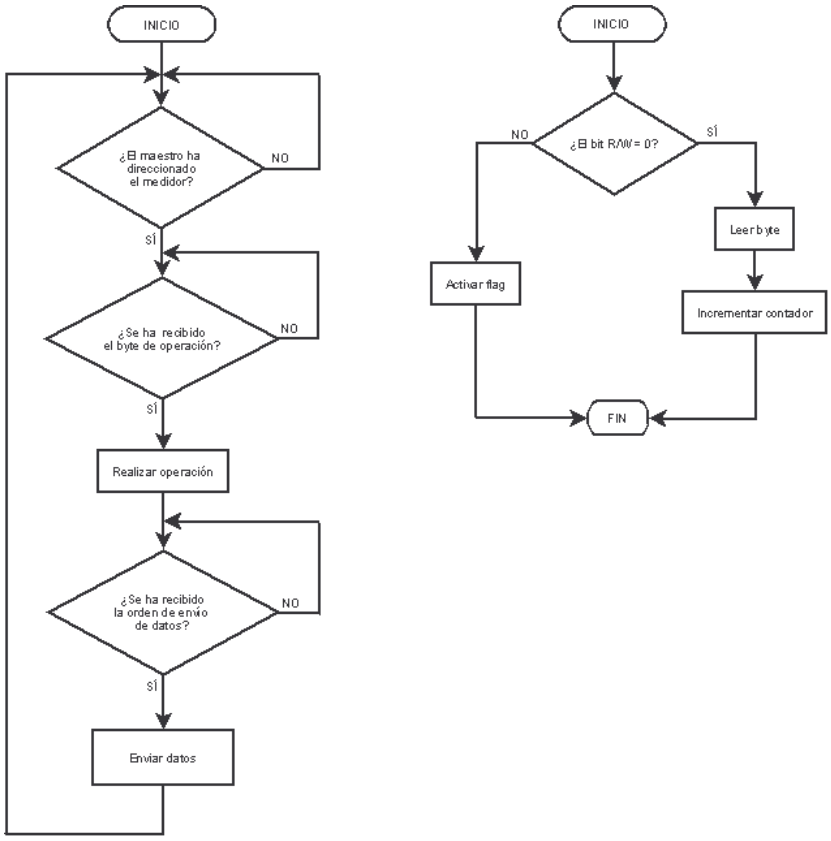


Ilustración 42: Diagramas de flujo de la función recepción/envío de datos de/a placa de control.

Cuando la placa de control escribe el byte de dirección en el bus, todos los esclavos conectados a él lo leen y el que tenga la misma dirección, queda direccionado por el maestro. Cuando el medidor ha sido direccionado, lee la operación, la realiza y espera para enviar los datos, a que el maestro se lo indique.

En la recepción ($R/\overline{W} = 0$), cada vez que se recibe un byte se produce una interrupción (diagrama de flujo de la derecha en la Ilustración 42), se lee el byte y se incrementa el valor de un contador inicializado en cero, cuando el contador alcanza el valor 2 quiere decir que el byte leído es la operación a realizar.

En la transmisión ($R/\overline{W} = 1$), se activa un flag que indica que el maestro quiere recibir datos por parte del esclavo. Se produce una interrupción por cada byte enviado.

3.2.1.2 Realizar operación:

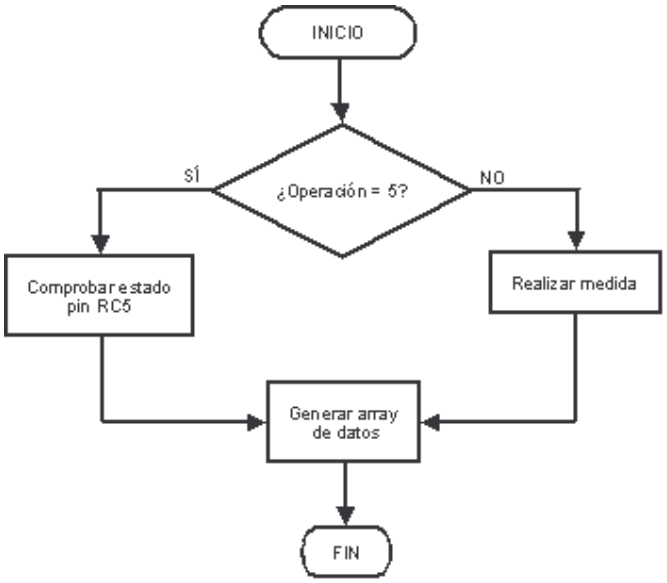


Ilustración 43: Diagrama de flujo de la función realizar operación.

Se pueden realizar dos tipos de operaciones en el medidor: disparo de los sensores de infrarrojos (según una secuencia determinada) y comprobar si al medidor se le ha asignado la puerta A o la puerta B dentro del sistema (operación sólo disponible en comunicación con la placa de control).

Las secuencias de disparo de los sensores pueden ser:

- Operación 1: Disparos consecutivos de izquierda a derecha, es decir, desde el primer sensor (1 ó 9) hasta el último (8 ó 16). Se obtienen 8 medidas por medidor.
- Operación 2: Disparos consecutivos de derecha a izquierda, es decir, desde el último sensor (8 ó 16) hasta el primero (1 ó 9). Se obtienen 8 medidas por medidor.
- Operación 3: Disparos de sensores alternos. Se obtienen 4 medidas por medidor.
- Operación 4: Disparo de un sensor cada cuatro. Si dividimos la base del robot en cuatro cuadrantes, con cuatro sensores por cuadrante, en esta operación se dispara uno por cuadrante hasta completar todos los sensores de la base. Se obtienen 8 medidas por medidor.

Los valores obtenidos por los sensores (DEC), en estas cuatro operaciones, se guardan en un array de datos de 8 bits. La estructura del array es:

- 1er byte: número de bytes de medida que se envían, es decir, este número será el mismo que el de sensores que se han disparado en la operación realizada.
- 2º byte y siguientes: valor de la medida (DEC) obtenida en el disparo de los sensores. Una medida por byte.
- Último byte: valor de comprobación, se incluye este código, para verificar que los datos se han enviado y recibido de forma correcta. Este valor se obtiene de sumar todos los bits de los datos que tengan el valor lógico '1'.

Por ejemplo, se supone que se realiza la operación 1 y que los datos proporcionados por los sensores son: 125, 200, 168, 75, 92, 88, 145, 178, la trama de datos sería:

8	125	200	168	75	92	88	145	178	30

En cambio, si la operación realizada fuese la cuatro y los datos proporcionados por los sensores fuesen: 125, 200, 168, 75, la trama de datos sería:

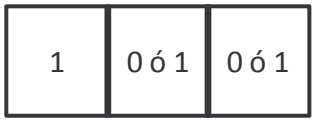
4	125	200	168	75	16

- Operación 5: Indica a la placa de control si el medidor es la puerta A o B en función del estado (conectado/desconectado) del jumper colocado en el pin RC5. Los datos se almacenan, de la misma forma, en un array de datos de 8 bits, pero éste está formado únicamente por tres bytes. La estructura del array es:

- 1er byte: número de bytes de que se envían, es decir, en este caso, este byte siempre será igual a uno.
- 2º byte: valor del estado (valor lógico '1' (jumper desconectado) ó '0' (jumper conectado)) del pin RC5.

- 3er byte: valor de comprobación, se incluye este código, para verificar que los datos se han enviado y recibido de forma correcta. Este valor se obtiene de sumar todos los bits de los datos que tengan el valor lógico ‘1’.

La trama de datos sería:



3.2.1.3 Realizar medida:

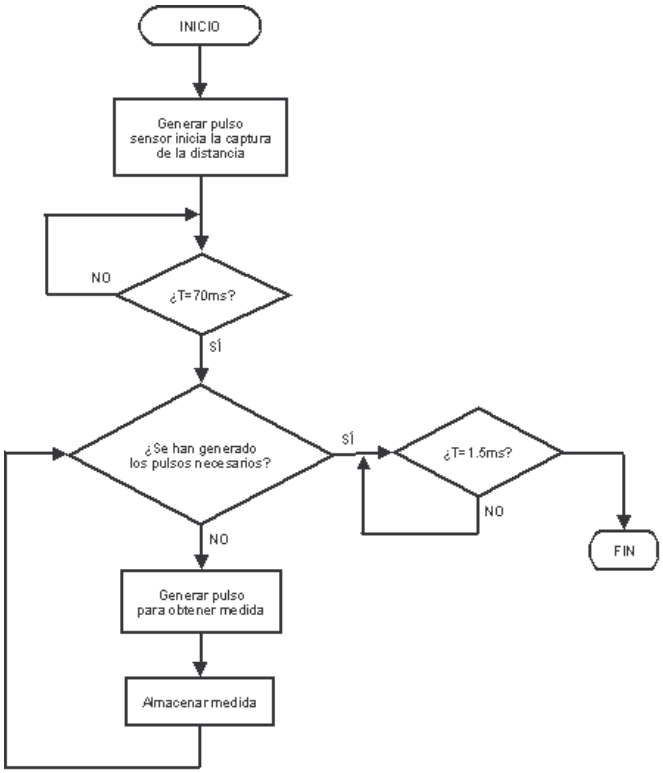
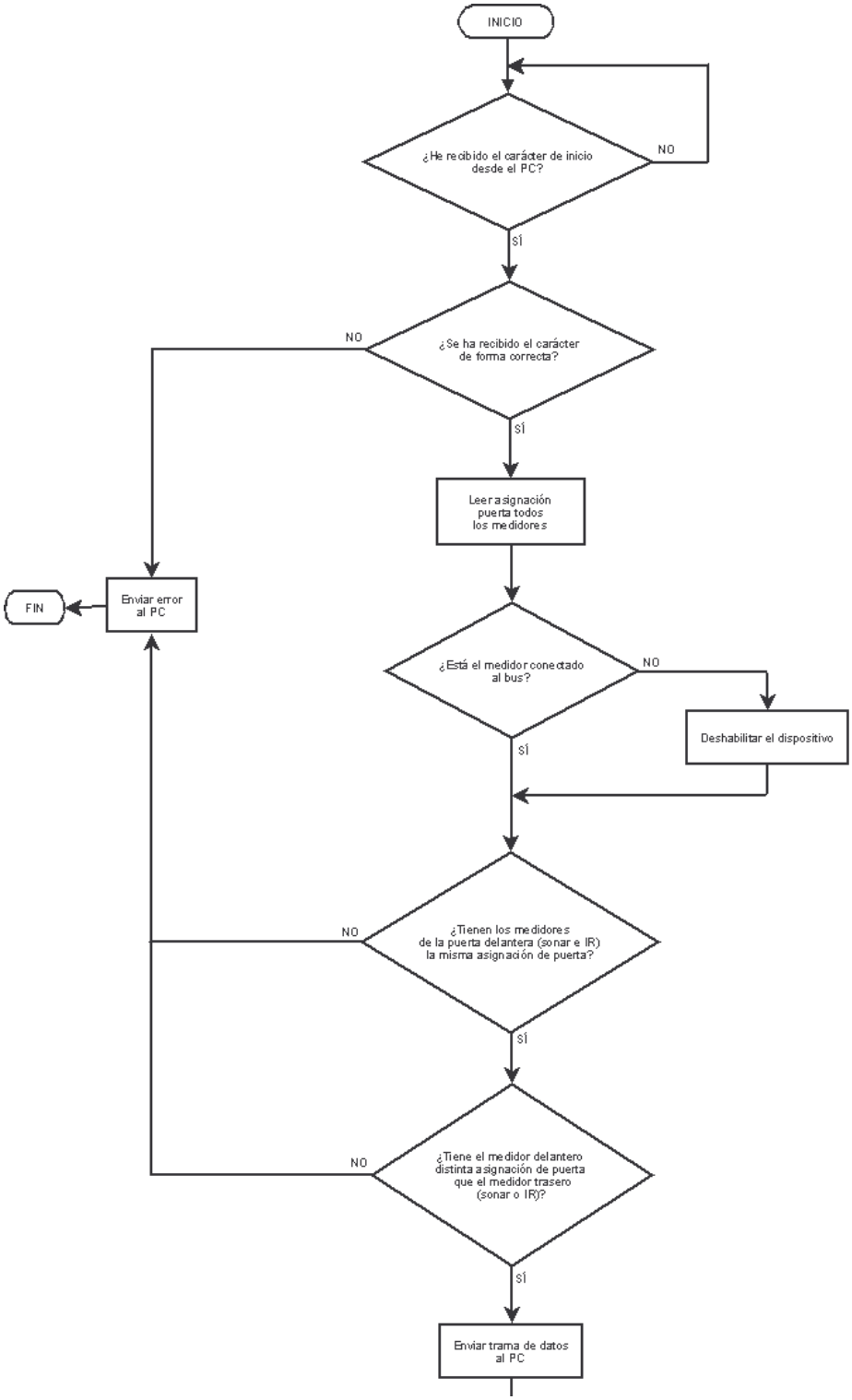


Ilustración 44: Diagrama de flujo de la función realizar medida.

Para obtener la medida de los sensores se sigue la secuencia establecida en el apartado 2.2.1.2, generando los pulsos necesarios y períodos de espera, mediante la utilización del timer 0 del PIC16F876.

Se pone a nivel lógico '1' el pin del micro que se corresponde con la entrada Vin del sensor, seguidamente se pone a nivel lógico ´0´esta misma entrada y se da inicio a la medida por parte del sensor. Para obtener el valor del sensor se generan ocho pulsos en la entrada Vin, obteniendo por cada pulso un bit de la medida (en el pin del micro que se corresponde con Vout del sensor). Finalmente hay que esperar un tiempo para que el sensor pueda realizar una nueva medida.

3.2.2 Placa de control



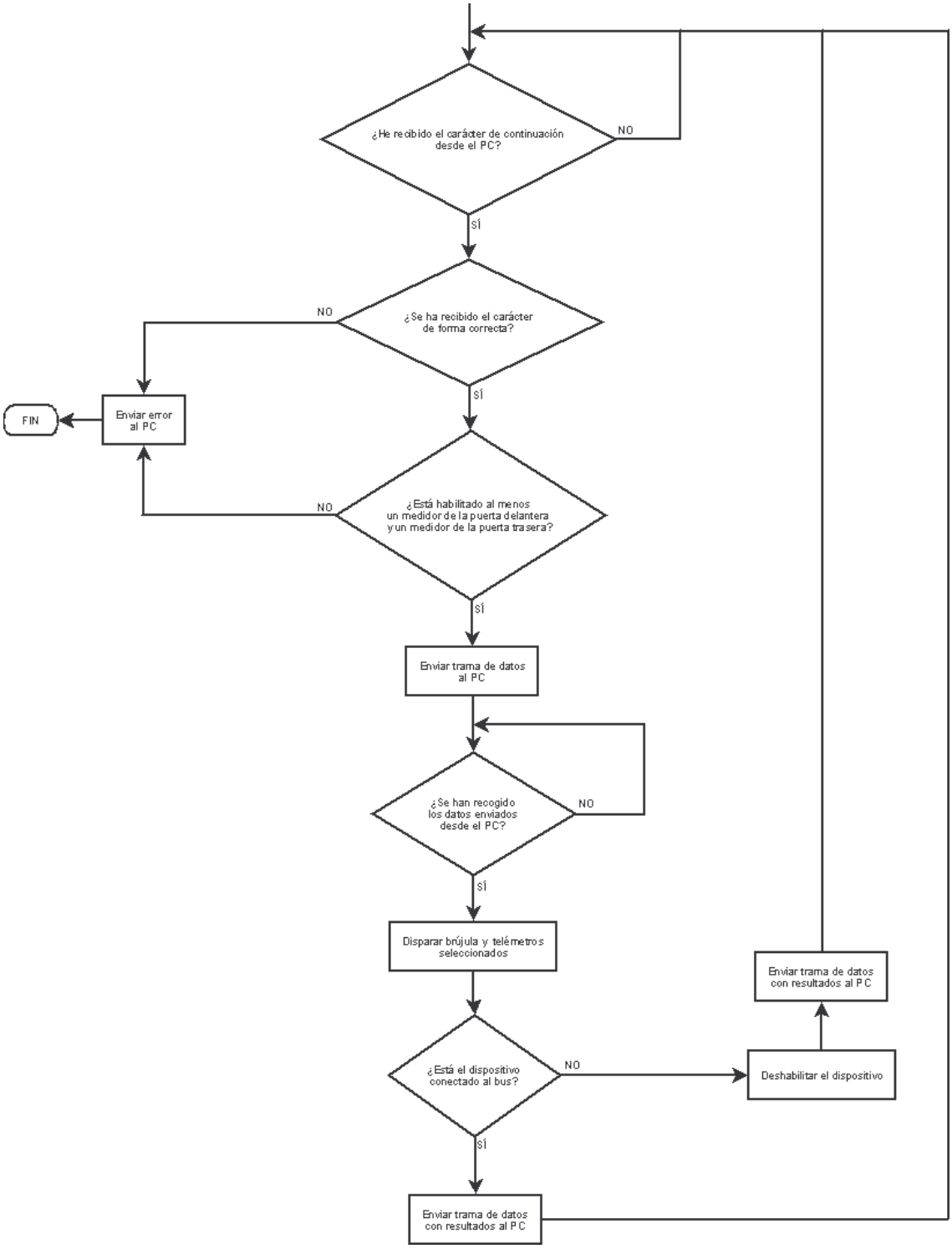


Ilustración 45: Diagrama de flujo programa placa de control.

La placa de control va establecer una comunicación con el PC del robot, del que recibe órdenes, y con los telémetros y brújula digital a los que le manda realizar operaciones.

Se definen en primer lugar todas las opciones de configuración, como el reset, la frecuencia de reloj y los registros del PIC, así como los parámetros de los módulos USART (velocidad de transmisión, pin Tx, pin Rx) e I^2C (dirección esclavo, velocidad de transmisión normal, pin SCL, pin SDA), se deshabilitan las interrupciones y se configuran los pines de los puertos B y C como entradas o salidas según corresponda.

Se definen las direcciones de los esclavos conectados al bus, en este caso:

- Telémetro por infrarrojos delantero: 10h.
- Telémetro por infrarrojos trasero: 20h.
- Telémetro por ultrasonidos delantero: 30h.
- Telémetro por ultrasonidos trasero: 40h.
- Brújula digital: C0h (definido por el propio módulo CMPS03).

La placa de control permanece a la espera hasta que recibe un carácter de comienzo desde el PC del robot. Este carácter sirve para determinar la correcta transmisión y recepción de datos entre el PC y la paca de control.

El carácter enviado es ‘r’, si la placa de control leyese otro carácter, se activaría un error y el sistema se pararía hasta que el fallo se haya reparado. Si el carácter leído es correcto comenzaría la inicialización del sistema. Ésta consiste en identificar que dispositivos están conectados al bus antes de comenzar las medidas, para ello la placa de control manda a los telémetros realizar la operación 5 (configuración de puerta), de esta forma se comprueba:

- Que los medidores de una misma puerta, al menos uno, está activo. Cuando se produce un fallo interno en alguno de los medidores o de la brújula y la placa de control no puede contactar con ellos, se puede desconectar el elemento que falla del bus I^2C, con lo que queda deshabilitado (desactivado), pero el sistema sigue funcionando. En caso, de que el medidor de infrarrojos y el medidor de ultrasonidos de la misma puerta (delantera o trasera) queden deshabilitados, el sistema se para y genera un error.

- La asignación de puerta (A o B) para los medidores colocados en la puerta delantera y para los colocados en la puerta trasera. La asignación de puerta tiene que ser la misma para los dos tipos de medidores, es decir, las posibles configuraciones que se pueden establecer son:

TIPO / Colocación	Sonar	Infrarrojos	TIPO / Colocación	Sonar	Infrarrojos
Medidores puerta delantera	PUERTA A	PUERTA A	Medidores puerta trasera	PUERTA B	PUERTA B
	PUERTA B	PUERTA B		PUERTA A	PUERTA A

Tabla 4: Configuración de puerta de los medidores.

En caso de que no se dé ninguna de las configuraciones anteriores el sistema genera un error y se para.

- Que el medidor de la puerta delantera y el medidor de la puerta trasera de un mismo tipo de sensor (sonar o infrarrojos), tengan asignaciones de puerta diferentes, es decir, que uno de ellos sea la puerta A y el otro sea la puerta B. En caso contrario (los dos sean la puerta A o los dos sean la puerta B), se genera un error y el sistema se para.

Cuando el sistema se para, hay que solucionar el error que produjo el fallo y volver a resetear todas las placas para restablecer el funcionamiento normal del sistema.

Una vez terminado este proceso de inicio y reconocimiento, la placa de control permanece a la espera de recibir una orden de operación desde el ordenador central del robot. Los datos que se reciben son: sensor a disparar (sonar y/o IR), puerta (A o B) y la secuencia de disparo de los sensores (operación). Una vez recibidos estos datos se realizan las operaciones y se transmiten los resultados al PC. Se reportan no sólo los datos obtenidos por los telémetros y la brújula digital, sino también cualquier fallo que pueda producirse durante el funcionamiento del sistema. Las funciones que realiza la placa de control se detallan a continuación.

3.2.2.1 Recibir posición de la brújula:

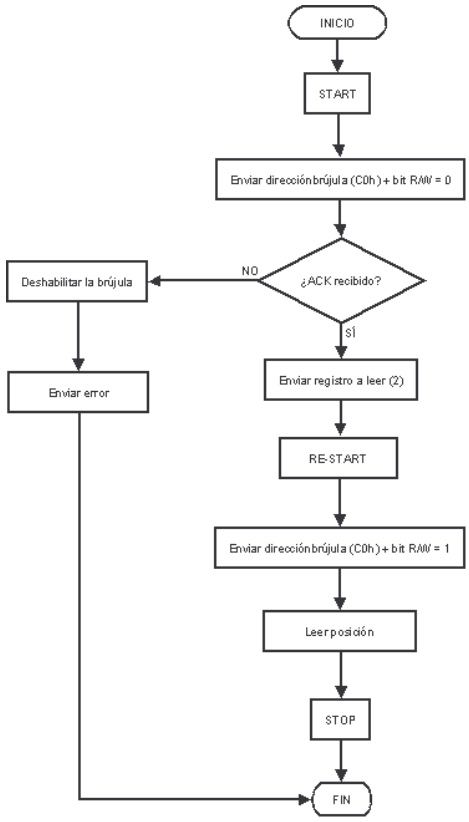


Ilustración 46: Diagrama de flujo de la función recibir posición de la brújula.

Para obtener la posición se lee el registro 2 del módulo CMPS03 mediante comunicación I^2C siguiendo las normas que establece el protocolo. Este registro almacena la posición en dos bytes (0-3599).

3.2.2.2 Envío/recepción de datos a/de los dispositivos esclavos:

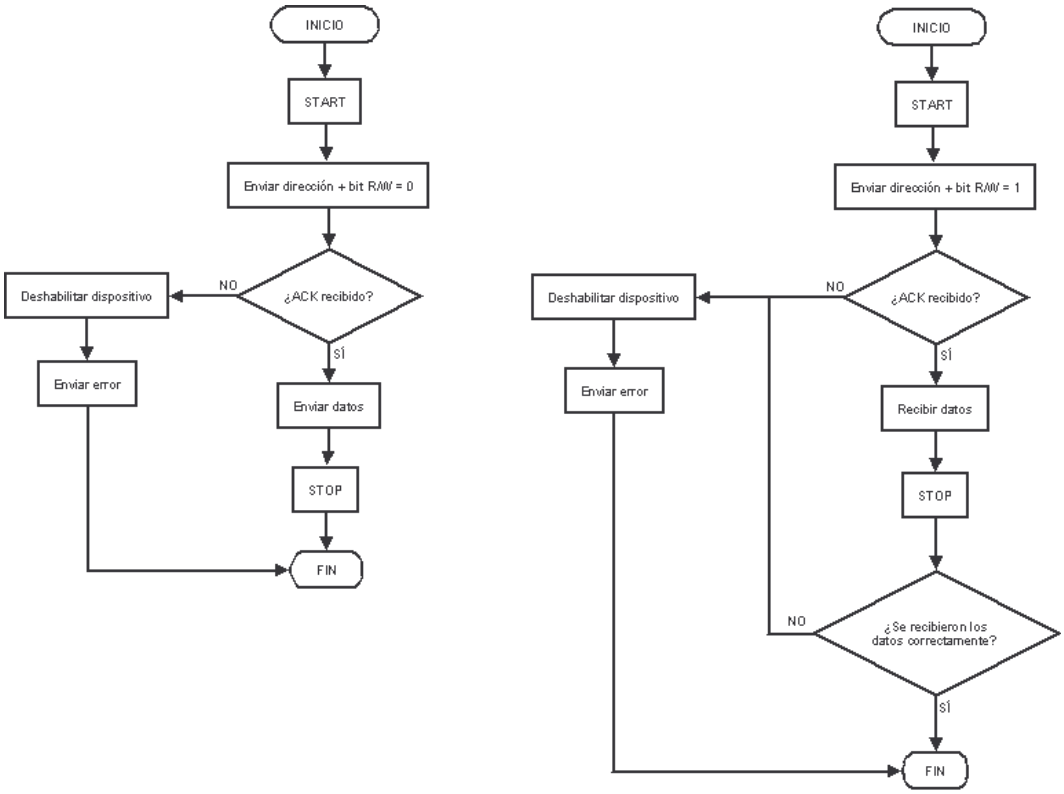


Ilustración 47: Diagramas de flujo de la función enviar/recibir datos a/de el esclavo.

Para poder enviar o recibir datos de/a el esclavo, se deben seguir las normas establecidas en el protocolo I^2C.

En el diagrama de flujo de la izquierda de la Ilustración 47, se puede observar la secuencia que hay que seguir para mandar datos al dispositivo esclavo. Si el dispositivo esclavo no responde o no está conectado al bus por cualquier motivo, se deshabilita, pero el sistema sigue funcionando, a no ser que estén deshabilitados los medidores de una misma puerta, es decir, medidores sonar e infrarrojos puerta delantera o medidores sonar e infrarrojos puerta trasera, en cuyo caso el sistema genera un error y se para.

En el diagrama de flujo de la derecha de la Ilustración 47, se puede observar la secuencia que hay que seguir para recibir datos desde el dispositivo esclavo. Al igual que en la transmisión, si al intentar comunicarse con el dispositivo esclavo, éste no responde o no está conectado al bus por cualquier motivo, se deshabilita, pero el sistema sigue funcionando, a no ser que estén deshabilitados los medidores de una misma

puerta, en cuyo caso el sistema genera un error y se para. Al recibir los datos se ha generado un sistema de comprobación para verificar que los datos se han recibido de forma correcta. El dispositivo esclavo cuenta el número de bits que tienen el nivel lógico '1' de los bytes de datos, esta información se transmite junto con los datos en el último byte. El maestro al recibir los datos cuenta el número de bits con nivel lógico'1' de los datos recibidos y este valor debe ser igual al valor del último byte recibido, si esto no fuese así se envía un error al PC, se deshabilita el medidor y no se recogerían las medidas correspondientes.

3.2.2.3 Envío de resultados al PC

Una vez que se realiza la operación especificada desde el PC, se envían los resultados obtenidos, siguiendo la siguiente secuencia:

1º → Array[1] de errores (17 datos de 8 bits). Cada dato contendrá el valor lógico '0' (si no hay fallo) o '1' (si hay fallo).

2º → Array de datos obtenidos de la brújula (2 datos de 8 bits).

3º → Array de datos obtenidos por el medidor mediante infrarrojos (9 datos de 8 bits).

4º → Array de datos obtenidos por el medidor mediante ultrasonidos (9 datos de 8 bits).

5º → Flag que indica si se han disparado los medidores de la puerta frontal o los de la puerta trasera.

3.2.2.3.1 Informe de errores.

Para detectar cualquier error que se pueda producir durante el funcionamiento del sistema, se crea un array de datos de 8 bits con 17 elementos, en el que cada uno representa un error. Este "informe de errores" es transmitido al PC. Los errores que se pueden producir son:

[1] En programación, conjunto o agrupación de variables del mismo tipo.

ERROR	DESCRIPCIÓN	ACCIÓN CORRECTIVA
Error [0]	No está activo, al menos un medidor de la puerta delantera.	El sistema se para. Arreglar, al menos uno y resetear el micro de la placa de control y reiniciar el sistema.
Error [1]	No está activo, al menos un medidor de la puerta trasera.	El sistema se para. Arreglar, al menos uno y resetear el micro de la placa de control y reiniciar el sistema.
Error [2]	Error en la transmisión de datos a los dispositivos esclavos (medidores y brújula).	Se inhabilita el dispositivo pero el sistema sigue funcionando. Para volver a habilitar el dispositivo en el sistema, una vez reparado, es necesario resetear el micro de la placa de control y reiniciar el sistema.
Error [3]	Error en la recepción de datos de los dispositivos esclavos (medidores y brújula).	Se inhabilita el dispositivo pero el sistema sigue funcionando. Para volver a habilitar el dispositivo en el sistema, una vez reparado, es necesario resetear el micro de la placa de control y reiniciar el sistema.
Error [4]	Error del código de comprobación en la recepción de datos de los medidores.	Se inhabilita el dispositivo pero el sistema sigue funcionando. Para volver a habilitar el dispositivo en el sistema, una vez reparado, es necesario resetear el micro de la placa de control y reiniciar el sistema.
Error [5]	Error en el medidor por infrarrojos puerta delantera.	Se inhabilita el dispositivo pero el sistema sigue funcionando. Para volver a habilitar el dispositivo en el sistema, una vez reparado, es necesario resetear el micro de la placa de control y reiniciar el sistema.
Error [6]	Error en el medidor por ultrasonidos puerta delantera.	Se inhabilita el dispositivo pero el sistema sigue funcionando. Para volver a habilitar el dispositivo en el sistema, una vez reparado, es necesario resetear el micro de la placa de control y reiniciar el sistema.

ERROR	DESCRIPCIÓN	ACCIÓN CORRECTIVA
Error [7]	Error en el medidor por infrarrojos puerta trasera.	Se inhabilita el dispositivo pero el sistema sigue funcionando. Para volver a habilitar el dispositivo en el sistema, una vez reparado, es necesario resetear el micro de la placa de control y reiniciar el sistema.
Error [8]	Error en el medidor por ultrasonidos puerta trasera.	Se inhabilita el dispositivo pero el sistema sigue funcionando. Para volver a habilitar el dispositivo en el sistema, una vez reparado, es necesario resetear el micro de la placa de control y reiniciar el sistema.
Error [9]	Diferente asignación de puerta a los medidores (sonar e infrarrojos) delanteros.	El sistema se para. Se deben colocar los jumpers correctamente, resetear el micro de la placa de control y reiniciar el sistema.
Error [10]	Diferente asignación de puerta a los medidores (sonar e infrarrojos) traseros.	El sistema se para. Se deben colocar los jumpers correctamente, resetear el micro de la placa de control y reiniciar el sistema.
Error [11]	Misma asignación de puerta a los medidores por infrarrojos delantero y trasero.	El sistema se para. Se deben colocar los jumpers correctamente, resetear el micro de la placa de control y reiniciar el sistema.
Error [12]	Misma asignación de puerta a los medidores por ultrasonidos delantero y trasero.	El sistema se para. Se deben colocar los jumpers correctamente, resetear el micro de la placa de control y reiniciar el sistema.
Error [13]	Error en la brújula digital.	Se inhabilita el dispositivo pero el sistema sigue funcionando. Para volver a habilitar el dispositivo en el sistema, una vez reparado, es necesario resetear el micro de la placa de control y reiniciar el sistema.
Error [14]	Error del código de comprobación en la recepción de los datos de los medidores por infrarrojos desde la placa de control-PC robot.	No se han transmitido o recibido los datos correctamente, por lo tanto, no se recogen las medidas del medidor por infrarrojos correspondiente.

ERROR	DESCRIPCIÓN	ACCIÓN CORRECTIVA
Error [15]	Error del código de comprobación en la recepción de los datos de los medidores por ultrasonidos desde la placa de control-PC robot.	No se han transmitido o recibido los datos correctamente, por lo tanto, no se recogen las medidas del medidor por ultrasonidos correspondiente.
Error [16]	Error en la comunicación PC robot-placa de control	No se pueden transmitir ni recibir datos a/de la placa de control. Se para el sistema.

Tabla 5: Informe de errores.

3.3 Implementación del driver (controlador).

Se han desarrollado funciones que permiten controlar el sistema de control sensorial desde el PC del robot. Como ya se ha explicado anteriormente la placa de control va conectada al puerto USB del PC y la comunicación se establece según el protocolo RS-232. Las funciones para manejar el puerto serie vienen definidas en la librería "termios.h".

3.3.1 Abrir y configurar puerto.

El PC del robot utiliza el sistema operativo Linux, en concreto, la distribución Fedora. Al igual que sucede en Windows, el sistema operativo Linux accede al puerto serie como si fuera un fichero, por lo que se utilizará la función *open* para abrir el puerto. Primero se tiene que determinar la dirección asignada a ese puerto, en Linux se consigue con el comando *dmesg*, esta dirección queda definida por la constante COM.

```
void configurar_puerto(struct termios *oldtio)
{
  struct termios newtio;

  fd=open(COM,O_RDWR | O_NOCTTY); //Apertura del puerto
  if(fd==-1)
  {
    perror("No se ha abierto COM");
    exit(-1);
  }

  tcgetattr(fd,oldtio);

  newtio.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
  newtio.c_cflag &= ~PARENB; //8N1
  newtio.c_cflag &= ~CSTOPB;
  newtio.c_cflag &= ~CSIZE;
  newtio.c_cflag |= CS8;
  newtio.c_oflag = 0;
  newtio.c_lflag = 0;

  newtio.c_cc[VTIME] = 0;
  newtio.c_cc[VMIN]  = 1;

  tcflush(fd, TCIFLUSH);
  tcsetattr(fd,TCSANOW,&newtio);

}
```

Ilustración 48: Función abrir y configurar puerto.

Como se puede observar en el código de la Ilustración 48, la función *open* tiene parámetros que permiten definir la forma de abrir el puerto. Con el parámetro "COM" se define la dirección asignada para el puerto y la variable "fd" queda definida como identificadora del puerto. Si no se puede abrir el puerto se muestra por pantalla el error.

El resto de funciones sirven para configurar el puerto, en este caso lo definimos para recibir/transmitir a 9600 baudios y 8N1, es decir, 8 bits de datos, sin paridad y un bit de stop.

3.3.2 Enviar cadena.

```
void enviar_cadena (unsigned char *opcion)
{
  int longitud=1;

  write(fd,opcion,longitud);
}
```

Ilustración 49: Función enviar cadena.

Para transmitir datos por el puerto serie se utiliza la función *write*, teniendo como parámetros el identificador del puerto serie, los datos a enviar y el número de bytes que se envían. Esta función devuelve el número de datos escritos o un valor -1 si los datos no se han transmitido correctamente.

3.3.3 Recibir cadena.

```
void recibir_cadena (void)
{
  int i;

  for(i=0;i<=16;i++)
  {
    robot.error[i]=0;
  }
  for(i=0;i<=1;i++)
  {
   robot.orientacion[i]=0;
  }
  for(i=0;i<=8;i++)
  {
    robot.medida_IR[i]=0;
  }
  for(i=0;i<=8;i++)
  {
    robot.medida_sonar[i]=0;
  }

  for(i=0;i<=16;i++)
  {
    read(fd,&robot.error[i],1);
  }
  for(i=0;i<=1;i++)
  {
   read(fd,&robot.orientacion[i],1);
  }
  for(i=0;i<=8;i++)
  {
    read(fd,&robot.medida_IR[i],1);
  }
  for(i=0;i<=8;i++)
  {
    read(fd,&robot.medida_sonar[i],1);
  }
  read(fd,&robot.flag_puerta_delantera,1);
  read(fd,&robot.flag_puerta_trasera,1);
  read(fd,&robot.k,1);
}
```

Ilustración 50: Función recibir cadena.

Para recibir datos se utiliza la función *read* que, al igual que la función anterior, tiene como parámetros el identificador del puerto, la dirección de memoria donde van a ser almacenados los datos recibidos y el número de bytes que se tienen que leer.

3.3.4 Obtener orientación y distancia al objeto.

La placa de control envía los resultados recibidos por los medidores y la brújula al PC. En el caso del medidor por infrarrojos se envían los valores DEC proporcionados por los sensores, en el caso de la brújula se envían los valores del registro 2 y en el caso del medidor por ultrasonidos se envían los tiempos de envío y recepción de la señal ultrasónica. Es en el PC donde se deben realizar las operaciones necesarias para conocer las distancias a los objetos y la orientación del robot. De esta forma se definen las siguientes funciones:

3.3.4.1 Distancia a los sensores infrarrojos:

El valor proporcionado por los sensores infrarrojos es transmitido al PC en 1 byte. Si el valor es menor de 80 se realiza una indicación de que el objeto se encuentra situado a más de 80cm del sensor y si el valor es mayor a 80 se calcula la distancia a través de la fórmula del apartado 2.2.1.3.

3.3.4.2 Distancia a los sensores ultrasonidos:

El valor proporcionado por los sensores de ultrasonidos es transmitido al PC en 1 byte. Pasa saber la distancia se tiene que multiplicar este valor por 2,72 [14].

3.3.4.3 Orientación de la brújula:

Como el valor de la brújula es transmitido en 2 bytes, se crea un puntero a un entero y el valor de éste será el contenido almacenado en los dos bytes. Al dividirlo entre 10, se obtiene el valor de la orientación del robot. Una vez obtenido se muestra por pantalla el resultado (Ilustración 51).

```
void obtener_valor_brujula (void)
{
        int *p;
        float orienta;

        p=&posicion;
        *p=robot.orientacion[1];
        p++;
        *p=robot.orientacion[0];
        orienta=posicion*0.1;
        printf("\r\nEl robot esta orientado %3.1f Norte\n",orienta);
}
```

Ilustración 51: Función para obtener la orientación de la brújula.

3.3.5 Cálculo código de comprobación de datos sensores infrarrojos y sonar.

Una vez que se reciben en el PC del robot los datos desde la placa de control, se tiene que hacer una comprobación para determinar que los datos se han recibido correctamente. Para ello se define una función de código de comprobación, en la que se establece el número de "unos" que existen en los datos recibidos y este valor tiene que coincidir con el valor del último byte recibido desde la placa de control.

```
void calculo_CRC_datos_IR (void)
{
   unsigned char num_bits_1_IR=0;
   int j,i;

   if(i2c_datos==0x03)
   {
        for(i=0;i<=3;i++)
        {
            for(j=0;j<=7;j++)
            {
               if((robot.medida_IR[i]>>j)&1)num_bits_1_IR++;
            }
        }
   }
   else
   {
      for(i=0;i<=7;i++)
      {
         for(j=0;j<=7;j++)
         {
               if((robot.medida_IR[i]>>j)&1)num_bits_1_IR++;
         }
      }
   }
   CRC_datos_IR=num_bits_1_IR;
}
```

Ilustración 52: Función cálculo código comprobación datos infrarrojos.

```
void calculo_CRC_datos_sonar (void)
{
   unsigned char num_bits_1_sonar=0;
   int j,i;

   if(i2c_datos==0x03)
   {
        for(i=0;i<=3;i++)
        {
            for(j=0;j<=7;j++)
            {
               if((robot.medida_sonar[i]>>j)&1)num_bits_1_sonar++;
            }
        }
   }
   else
   {
        for(i=0;i<=7;i++)
        {
           for(j=0;j<=7;j++)
           {
              if((robot.medida_sonar[i]>>j)&1)num_bits_1_sonar++;
           }
        }
   }
   CRC_datos_sonar=num_bits_1_sonar;
}
```

Ilustración 53: Función cálculo código de comprobación datos sonar.

3.3.6 Cerrar puerto.

```
void cerrar_puerto(struct termios *oldtio)
{
  tcsetattr(fd,TCSANOW,oldtio);
  close(fd);

}
```

Ilustración 54: Función cerrar puerto.

Una vez recibidos todos los datos se tiene que cerrar el puerto. Para ello se utiliza la función *close*, estableciendo como parámetro el identificador del puerto.

3.3.7 Programa de prueba.

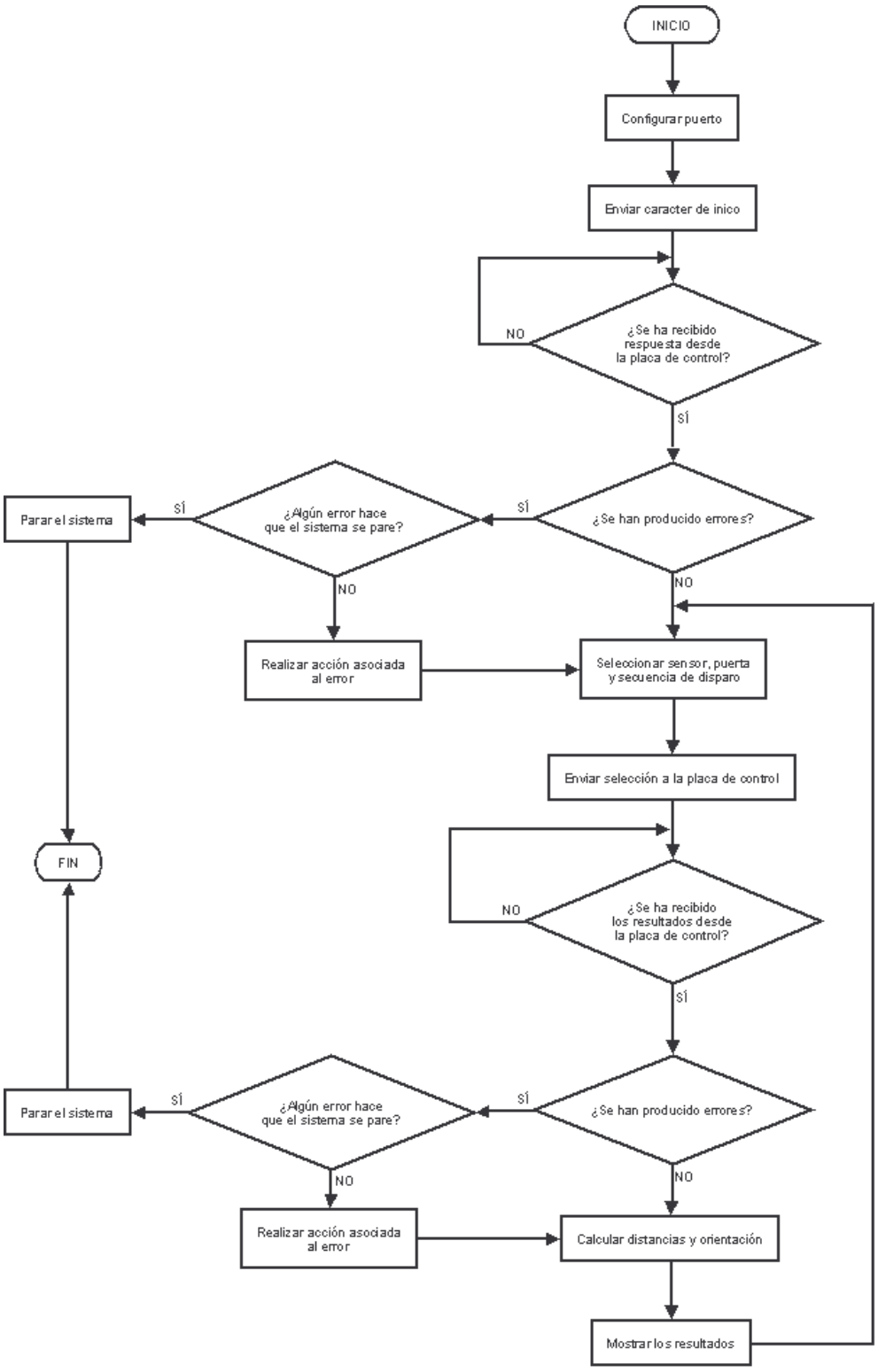


Ilustración 55: Diagrama de flujo del programa de prueba.

En primer lugar se definen los campos de la estructura donde van a ser almacenados los datos recibidos desde la placa de control. Posteriormente se definen las funciones descritas en el apartado anterior y que van a permitir controlar la placa desde el PC del robot.

Se envía el carácter de inicio desde el PC a la placa para verificar la perfecta comunicación entre ambos dispositivos. Se espera a que la placa envíe todos los datos de inicialización: dispositivos conectados al bus, errores producidos… Se verifica si hay errores, se muestran por pantalla y si hay alguno crítico se para el sistema. Los errores que se consideran críticos son: los telémetros de la puerta delantera o de la puerta trasera tienen diferente asignación de puerta, por ejemplo, el telémetro por IR delantero tiene asignada la puerta A y el telémetro por ultrasonidos delantero tiene asignada la puerta B; un mismo tipo de telémetro tiene la misma asignación en la puerta delantera y en la puerta trasera, por ejemplo, telémetro por ultrasonidos delantero tiene asignada la puerta A y telémetro por ultrasonidos trasero también tiene asignada la puerta A; otro error que se considera crítico es que los dos telémetros de una puerta no estén conectados al bus, por ejemplo, desconectados el telémetro por IR delantero y el telémetro por ultrasonidos delantero.

La fase de inicialización sólo se realiza una vez durante todo el funcionamiento del sistema y una vez realizada se pasa a la toma de medidas. Se tiene que elegir la puerta que se dispara, el sensor y la secuencia de disparo, estos datos se envían a la placa que efectúa las operaciones indicadas y devuelve los resultados al PC. Si durante el proceso se produce algún error éste es comunicado de inmediato al PC y si se produce algún error crítico se para el sistema. En todo momento cuando se reciben datos, ya sean errores o distancias se muestran por pantalla.

Cuando los datos recibidos son los valores obtenidos por los sensores (IR, ultrasonidos y brújula), se calcula la distancia a los objetos y la orientación del robot según se explica en el apartado 3.3.4, también se incluye el código de comprobación que verifica que los datos enviados por la placa y los recibidos en el PC son iguales (apartado 3.3.5).

CAPÍTULO 4.- RESULTADOS EXPERIMENTALES.

En este capítulo se van a detallar todas las pruebas realizadas a los componentes del sistema y los resultados obtenidos.

4.1 Telémetro por infrarrojos.

4.1.1 Calibración del sensor.

Antes de poder realizar medidas con el sensor, se tiene que determinar su rango de medida. Según las especificaciones del fabricante éste tiene que ser de 10-80cm. Para determinar el rango de medida del sensor se coloca un objeto a diferentes distancias conocidas, obteniendo los valores DEC[1] para cada una de ellas. Los resultados obtenidos son los que se muestran a continuación:

OBJETO: CAJA	
MATERIAL: CARTÓN	
COLOR: VERDE	
L real (cm)	DEC
4	131
5	179
6	210
7	238
8	236
9	227
10	214
15	165
18	148
20	140
23	130
24	127
25	124
30	114
35	107
40	101
45	97
50	93
55	90
60	88
65	85
70	83
75	81
80	78
85	76
90	75
95	74
100	73

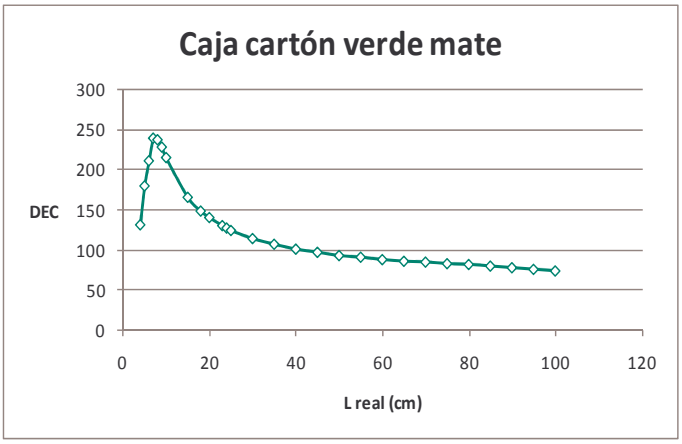


Ilustración 56: Curva DEC vs Lreal obtenida experimentalmente.

[1] Valor que proporciona el sensor y que tiene relación con el ángulo que forma la luz incidente con la luz reflejada y por tanto con la distancia al objeto.

Estos resultados son los esperados según la hoja de características del sensor y la curva obtenida experimentalmente es similar a la gráfica teórica proporcionada por el fabricante del sensor (ANEXO A).

Como se puede observar en la Ilustración 56, el sensor, en un principio, podría tener un rango de medida de 4-100cm. El valor proporcionado por el sensor (DEC), cuando el objeto está muy próximo a él, se sitúa en torno a un valor de 130, luego según se va alejando el objeto, este valor aumenta hasta llegar a un máximo de 238 que se corresponde con una distancia real de 7cm. A partir de este punto de inflexión se produce un descenso rápido del valor DEC respecto a la distancia, es decir, un pequeño incremento en la distancia real al objeto produce un gran decremento en el valor proporcionado por el sensor. Al llegar a una distancia alrededor de los 60cm el descenso de DEC se produce de forma más pausada, es decir, una variación grande en la distancia hace variar muy pocas unidades el valor del sensor. A partir de 80 cm la variación del valor de DEC es prácticamente constante, de manera que no se puede obtener con precisión la distancia a la que se encuentra el objeto, se hará una indicación de objeto detectado a +80cm.

La fórmula que relaciona DEC con L se establece a partir del punto máximo de salida del sensor, es decir, a partir de los 7cm, se puede concluir entonces que el rango de medida, en un principio, comprende desde los 7cm hasta los 80 cm.

Se puede observar, por el gráfico obtenido, una primera limitación en la toma de medidas del sensor, hay valores DEC que se corresponden con dos distancias reales, por ejemplo, si el sensor proporciona un valor de 170 no se sabe si el objeto se encuentra situado a una distancia de 5cm o de 15cm. Esto sucede en el intervalo comprendido entre los 4 y 25cm (valores DEC mayores a 125). Todo esto puede hacer que el objeto esté más cerca del sensor de lo que en realidad se indica, sobre todo si el objeto no ha sido detectado por el sensor antes de entrar en este intervalo. Por este motivo se recomienda hacer aproximaciones progresivas a los objetos y si no es posible, el robot dispone en su base de bumpers que minimizan el riesgo de daños en el mismo, cuando choque contra un objeto en caso de una mala interpretación de la distancia por parte del sensor (cuando hay un contacto de un objeto con el bumper se para el robot).

Durante las pruebas en el laboratorio se pudo observar que cuando no había objeto colocado delante del sensor, algunos sensores daban valores en torno a 100 y

otros daban valores menores de 80 (resultado esperado en todos los casos), después de muchas investigaciones se comprobó que la carcasa de plástico del sensor tenía conductividad y era necesario que estuviera conectada a tierra para que las medidas fueran correctas. Por este motivo los sensores que estaban bien conectados con la puerta de la base del robot daban buenas medidas, mientras que los que no tenían buen contacto daban falsas medidas. Este es un dato que hay que tener muy en cuenta, porque el sensor puede detectar un objeto situado a 40cm aproximadamente (DEC = 100), cuando en realidad no hay ningún objeto situado delante de él.

4.1.2 Efecto de la reflectividad en la toma de medidas.

Se han realizado diferentes medidas sobre objetos de material y color diferente para estudiar el efecto de la reflectividad sobre el valor proporcionado por el sensor y en consecuencia con la distancia medida. La reflectividad va a hacer que se produzcan variaciones del valor DEC cuando se efectúan medidas sobre diferentes objetos situados a una misma distancia. Si no hubiese influencia, el valor DEC debería ser el mismo para todos los objetos. Se han escogido para realizar el estudio tres puntos de la curva de transferencia del sensor, 30, 60 y 80cm.

DISTANCIA REAL AL OBJETO = 30cm			
Material	Color	Forma	DEC
Metal	Gris	Cuadrada	111
Poliespan	Blanco	Cuadrada	113
Metal	Negro	Cilíndrica	98
Cartón	Amarillo brillante	Cuadrada	108
Cartón	Azul brillante	Cuadrada	112
Cartón	Rojo brillante	Cuadrada	105
Tela	Negro	Cuadrada	114
Yeso	Blanco	Cuadrada	114
Cartón	Morado	Rectangular	114
Cartón	Blanco	Rectangular	115
Cartón	Verde	Rectangular	114
Cartón	Naranja fosforito	Rectangular	114
Cartón	Azul	Rectangular	114
Cartón	Negro	Rectangular	114
Cartón	Amarillo	Rectangular	114
Sin objeto			<80
DEC medio			114

Ilustración 57: Valores DEC para L=30cm.

DISTANCIA REAL AL OBJETO = 60cm			
Material	Color	Forma	DEC
Metal	Gris	Cuadrada	84
Poliespan	Blanco	Cuadrada	90
Metal	Negro	Cilíndrica	78
Cartón	Amarillo brillante	Cuadrada	85
Cartón	Azul brillante	Cuadrada	86
Cartón	Rojo brillante	Cuadrada	81
Tela	Negro	Cuadrada	89
Yeso	Blanco	Cuadrada	88
Cartón	Morado	Rectangular	88
Cartón	Blanco	Rectangular	87
Cartón	Verde	Rectangular	88
Cartón	Naranja fosforito	Rectangular	88
Cartón	Azul	Rectangular	88
Cartón	Negro	Rectangular	88
Cartón	Amarillo	Rectangular	88
Sin objeto			<80
DEC medio			88

Ilustración 58: Valores DEC para L=60cm.

DISTANCIA REAL AL OBJETO = 80cm			
Material	Color	Forma	DEC
Metal	Gris	Cuadrada	78
Poliespan	Blanco	Cuadrada	83
Metal	Negro	Cilíndrica	75
Cartón	Amarillo brillante	Cuadrada	79
Cartón	Azul brillante	Cuadrada	79
Cartón	Rojo brillante	Cuadrada	75
Tela	Negro	Cuadrada	84
Yeso	Blanco	Cuadrada	82
Cartón	Morado	Rectangular	83
Cartón	Blanco	Rectangular	82
Cartón	Verde	Rectangular	82
Cartón	Naranja fosforito	Rectangular	82
Cartón	Azul	Rectangular	83
Cartón	Negro	Rectangular	83
Cartón	Amarillo	Rectangular	82
Sin objeto			<80
DEC medio			82

Ilustración 59: Valores DEC para L=80cm.

Según los resultados obtenidos se pueden obtener varias conclusiones. Se puede destacar que el efecto de la reflectividad del objeto hace que el valor DEC varíe en varias unidades, por lo tanto, tendrá menor influencia, en la toma de medidas, cuanto más cerca esté el objeto al sensor, porque es en esta zona donde las variaciones de DEC provocan una menor variación de la distancia, por lo que el error cometido será menor.

Otro aspecto a tener en cuenta es que los objetos tienen que tener una superficie plana en donde incida la luz infrarroja, porque en caso contrario no se obtiene la precisión deseada. Esto se puede apreciar al observar en la tabla, por ejemplo, de los valores DEC para L=30cm, cuando el objeto es cilíndrico el valor DEC varía bastante respecto al valor medio obtenido con los otros objetos. De la misma manera el objeto debe tener una altura mínima de 15cm para que pueda ser detectado por el sensor.

En concordancia con las especificaciones del fabricante, se comprueba que no existe influencia debida al color del objeto, incluso con el color negro se obtienen los mismos resultados que para el resto de colores, así como con colores fosforescentes. Lo que sí tiene influencia es el material, no se obtienen los mismos resultados si el material es cartón, metal, madera… También se ha comprobado que la luminosidad del entorno donde se realizan las medidas tiene influencia a la hora de conseguir una mayor precisión.

Se ha comprobado que los objetos que tienen un color brillante pueden provocar "falsos reflejos" que hagan que el valor de DEC no sea el que le corresponde teniendo en cuenta la distancia del objeto al sensor. Se detectan los objetos a menor o mayor distancia de la que en realidad están, situación potencialmente peligrosa en el último caso, pero el robot dispone en su base de bumpers que minimizan el riesgo de daños en el mismo cuando se produce un choque contra un objeto.

Algo parecido sucede cuando la luz infrarroja incide sobre una superficie esquinada, esto puede hace que el sensor crea que el objeto está más cercano de lo que en realidad está (valor DEC más alto de lo que debería ser), para evitar este efecto la colocación óptima de estos sensores sería de forma vertical. Esto no se puede llevar a cabo porque, como ya se explicó, los sensores ya vienen integrados en posición horizontal en la base del robot.

4.1.3 Cálculo del valor de las constantes Kg y Ko.

Como ya se comentó en el apartado 2.2.1, antes de poder realizar medidas con el sensor, se tiene que calcular el valor de las constantes Kg y Ko. Se tomarán como datos para el cálculo, los valores DEC medios así como las distancias a los objetos, se necesitan dos puntos de la curva [12].

Para L_1=30cm ⇒ DEC_1=114; L_2=60cm ⇒ DEC_2=88.

Despejando Kg de la fórmula que relaciona DEC y L, para los dos puntos que se consideran se obtiene el valor de Ko.

$$Ko = \frac{(L_2 * DEC_2 - L_1 * DEC_1)}{L_2 - L_1} = \frac{(0.6 * 88) - (0.3 * 114)}{0.6 - 0.3} = 62$$

Despejando Ko de la fórmula que relaciona DEC y L, para los dos puntos que se consideran se obtiene el valor de Kg.

$$Kg = \frac{(DEC_2 - DEC_1) * L_2 * L_1}{L_1 - L_2} = \frac{(88 - 114) * 0.6 * 0.3}{0.3 - 0.6} = 15.6$$

De esta forma el valor de L queda determinado como:

$$L(cm) = \frac{15.6}{DEC - 62} * 100$$

4.1.4 Realización de medidas.

Una vez determinada la relación que existe entre el valor proporcionado por el sensor (DEC) y la distancia real (L) al objeto se pueden realizar medidas para comprobar el correcto funcionamiento y estimar el error cometido.

Se colocan tres objetos en diferentes sensores a diferentes distancias, tal y como se muestra en la Ilustración 60, y se efectúa la secuencia de disparo de sensores alternos (operación 3), obteniendo los resultados que se exponen a continuación de la ilustración siguiente:

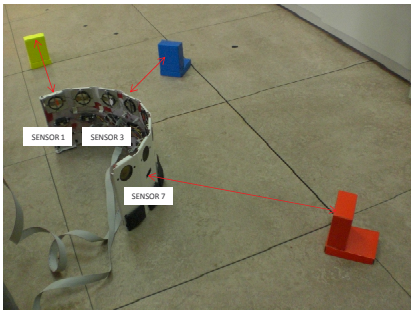


Ilustración 60: Realización de medidas con objetos en sensores 1,3 y 7.

SENSOR	OBJETO			L real (cm)	DEC	Lmedida (cm)	Error absoluto (cm)	Error (%)
	Material	Color	Forma					
MEDIDA 1								
1	Cartón	Amarilllo brillante	Cuadrada	60	86	65,00	5,00	8,33
3	Cartón	Azul brillante	Cuadrada	43,5	96	46,40	2,90	6,67
5	Cartón	Rojo brillante	Cuadrada	39,8	98	43,70	3,90	9,80
7	Sin objeto	-	-	-	77	+80	-	-
MEDIDA 2								
1	Cartón	Gris	Rectangular	50	93	50,32	0,32	0,65
3	Cartón	Blanco	Rectangular	25	125	24,76	0,24	0,95
5	Metal	Gris	Rectangular	18	147	18,35	0,35	1,96
7	Sin objeto	-	-	-	76	+80	-	-
MEDIDA 3								
1	Cartón	Amarilllo brillante	Cuadrada	31,5	117	28,36	3,14	9,96
3	Yeso	Blanco	Cuadrada	23	130	22,94	0,06	0,26
5	Metal	Gris	Cuadrada	35	108	33,91	1,09	3,11
7	Cartón	Negro	Rectangular	40	100	41,05	1,05	2,63
MEDIDA 4								
1	Cartón	Amarillo brillante	Rectangular	75	78	97,50	22,50	30,00
3	Cartón	Azul	Rectangular	75	81	82,11	7,11	9,47
5	Cartón	Blanco	Rectangular	75	80	86,67	11,67	15,56
7	Cartón	Negro	Rectangular	75	81	82,11	7,11	9,47
MEDIDA 5								
1	Cartón	Amarillo	Rectangular	60	89	57,78	2,22	3,70
3	Cartón	Azul	Rectangular	60	87	62,40	2,40	4,00
5	Cartón	Blanco	Rectangular	60	88	60,00	0,00	0,00
7	Cartón	Negro	Rectangular	60	87	62,40	2,40	4,00

Ilustración 61: Medidas realizadas con el telémetro de infrarrojos.

Se han realizado las medidas necesarias para comprobar el correcto funcionamiento del telémetro y sacar conclusiones respecto a los datos proporcionados por el sensor. La Lmedida (cm) es el dato obtenido por el hyperterminal del PC, es decir, es el resultado transmitido por el telémetro.

Se comprueba que se recogen de forma correcta todos los datos. En el rango de medida del sensor, se muestra por pantalla el resultado de la distancia y fuera del rango se hace una indicación de objeto a +80cm. Esto demuestra su correcto funcionamiento.

Medida 1: Se colocan tres cajas de colores brillantes diferentes para comprobar el efecto de la reflectividad en el rango de medida. Se puede observar que cuanto más cerca está el objeto del sensor menor es la influencia de la reflectividad y por tanto menor es el error cometido (en torno a los 4cm). Esto tiene una explicación lógica, si se observa la curva de la relación de DEC y L, en la que por debajo de los 60cm aproximadamente las variaciones grandes en el valor DEC (efecto que produce la reflectividad) no provocan desviaciones grandes de la medida L y el error cometido se reduce. También se comprueba que cuando no hay ningún objeto colocado delante del sensor, se realiza una indicación de que el objeto se encuentra situado a +80cm.

Medida 2: Se colocan tres cajas de colores diferentes pero no brillantes dentro del rango por debajo de los 60cm para comprobar el efecto de la reflectividad y la correcta linealización de la curva DEC vs L. Una de las cajas es de material diferente a las demás. Se puede observar que se obtienen valores bastante aceptables (por debajo de 1cm de error) cuando las cajas son de cartón y un error un poco mayor al cambiar la superficie y el material del objeto (variar la reflectividad), pero aceptable también (por debajo de 1cm de error). De la misma forma, que en la medida anterior, se comprueba que cuando no hay ningún objeto colocado delante del sensor, se realiza una indicación de que el objeto se encuentra situado a +80cm.

Medida 3: Se colocan cuatro cajas de diferentes colores, forma y material para estudiar los errores cometidos al variar estas características del objeto dentro del rango por debajo de los 60cm. Se puede observar que no hay prácticamente influencia ni del color (excepto si es brillante) ni del material y que se obtienen errores bajos y permitidos (en torno a 1cm). Si el objeto tiene un color brillante el error aumenta hasta los 3cm.

Medida 4: Se colocan cuatro cajas de la misma forma y material pero de colores diferentes, se pretende estudiar el efecto de la reflectividad por encima de los 60cm de distancia. Se puede comprobar que los errores obtenidos son mayores (por encima de los 7 cm) y que hay influencia según el color. Esto tiene una explicación lógica si se observa la curva DEC vs L en la que en la zona situada por encima de los 60cm cualquier variación por pequeña que sea del valor DEC provoca grandes variaciones en el valor de L, por lo que el error cometido al calcular la distancia será mayor. Los resultados obtenidos están en concordancia con los resultados teóricos suministrados por el fabricante del sensor.

Medida 5: Se colocan cuatro cajas de colores diferentes pero misma forma y material a 60cm de distancia. Se observa que los errores cometidos empiezan a ser elevados pero dentro del rango aceptable (en torno a 2,5cm). Y también se comprueba que no hay influencia debida al color del objeto.

Después de analizar los resultados experimentales se puede concluir que en el rango de 7-60cm se obtienen medidas bastante aceptables y sin influencia significativa de la reflectividad del objeto. Por encima de esta distancia la influencia es mayor y el error cometido puede variar en varios centímetros. Por todo lo explicado en este apartado y en el anterior se determina que el rango óptimo de medida es de 25-60cm.

4.1.4.1 Errores en la medida:

Como se puede observar la distancia a la que se encuentra el objeto es una medida indirecta que se obtiene a partir de otras medidas indirectas, como son el valor DEC y las constantes Kg y Ko. El cálculo de las constantes y del valor proporcionado por el sensor dependen de una medida directa que es la distancia real a la que se encuentra el objeto. Esta medida directa se obtiene mediante una regla de medir, es decir, ya introduce un error propio del instrumento y además otro error de colocación del observador para determinar la distancia. El error final será la propagación de este error, en todas las medidas que dependan de esta magnitud, sumado a la propia precisión del sensor. Con todo esto, que hay que tener en cuenta, se puede concluir que se obtienen errores aceptables en el rango de 25-60cm.

4.2 Brújula digital.

4.2.1 Calibración de la brújula.

Antes de calibrar la brújula, el módulo CMPS03, debe mantenerse perfectamente horizontal con los componentes hacia arriba y los dos sensores en la cara inferior. Hay que mantener el módulo alejado de objetos metálicos y muy especialmente de objetos magnéticos como imanes y altavoces. Para poder llevarla a cabo, es necesario conocer con precisión la dirección en la que se encuentran los cuatro puntos cardinales, esto se conoce por medio de una brújula magnética.

La calibración de la brújula digital puede hacerse por cualquiera de los siguientes dos métodos:

- El Método I^2C.

Consiste en escribir el valor decimal 255 en el registro 15 del módulo por cada uno de los cuatro puntos cardinales. El valor 255 es borrado internamente cada vez que se completa la calibración Los puntos de calibración pueden hacerse en cualquier orden, pero siempre es necesario calibrar los 4 puntos.

- El Método del pulsador.

Consiste en utilizar un pulsador entre masa y el pin 6 del circuito, con el fin de iniciar la calibración. Hay que tener en cuenta que este pin tiene una resistencia de polarización interna y puede dejarse sin conectar una vez realizada la calibración. Para realizarla, bastará con poner a masa el pin 6 momentáneamente por cada uno de los puntos cardinales. De igual forma que con el otro método, los puntos pueden calibrarse en cualquier orden, pero siempre es necesario calibrar los 4 puntos cardinales.

De los dos métodos posibles se ha elegido el primero, puesto que ya se tiene implementado el circuito para permitir su comunicación I^2C con el micro. De esta forma, se coloca el módulo orientado a cada uno de los cuatro puntos cardinales y, por cada uno, se escribe el número 255 en el registro 15.

4.2.2 Realización de medidas.

Una vez realizada la calibración, se realizan medidas para determinar si ésta se ha realizado de forma correcta y determinar el error en la medida proporcionada por el sensor. Los valores determinados por el sensor son comparados por los proporcionados por una brújula magnética. Se pueden observar los resultados a continuación, así como ilustraciones en las que se pueden ver como se realizaron las medidas.

Brújula magnética	Brújula digital
300º	300º
45º	47.2º
135º	135.4º

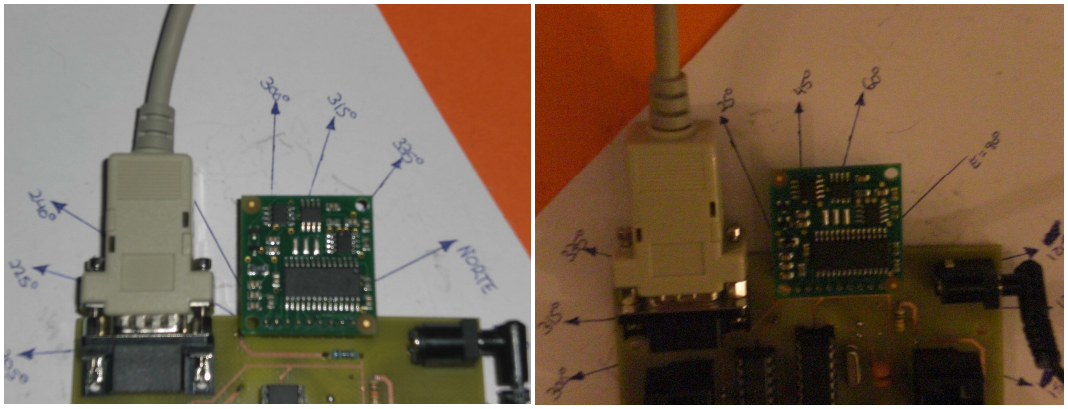


Ilustración 62: Medidas realizadas con la brújula digital.

Se comprueba que la especificación del fabricante es correcta. Se consigue una buena aproximación de la orientación real a la medida. Los posibles errores pueden ser derivados de la precisión de la propia brújula, tanto digital como analógica, así de la lectura que se realiza sobre la brújula magnética ya que ésta depende de la posición del observador y por tanto influye en el error.

4.3 Sistema general.

Antes de probar el funcionamiento del sistema general, se tiene que calibrar el telémetro por ultrasonidos para establecer el rango de medida real. Se realizan disparos consecutivos sobre un objeto colocado a diferentes distancias obteniendo un rango de medida de 45cm-10m.

Para realizar las medidas del sistema completo se conectan todos los dispositivos a la placa de control y ésta a su vez se conecta al PC del robot. Se colocan objetos en la parte frontal y trasera del robot a diferentes distancias, para que puedan ser detectados por los sonar y por los infrarrojos. Todos los telémetros han sido configurados con las mimas secuencias de disparo. Se deja sin conectar los sonar de la puerta trasera para verificar el correcto funcionamiento del sistema.

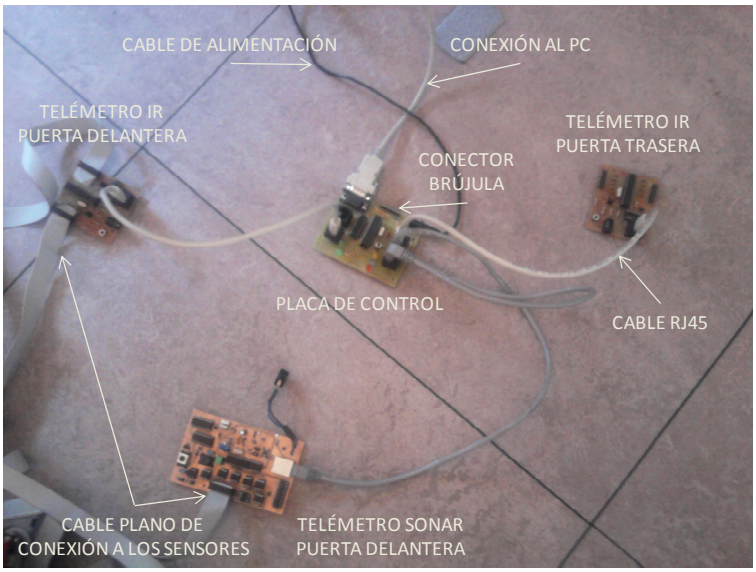


Ilustración 63: Esquema de conexiones del sistema general.

Al ejecutar el driver, se inicializa el sistema y se muestra el siguiente error por pantalla:

```
   ERROR AL TRANSMITIR DATOS

EL MEDIDOR POR ULTRASONIDOS DE LA PUERTA TRASERA NO SE ENCUENTRA
DISPONIBLE. SE INHABILITA EL DISPOSITIVO.
```

Resultado esperado porque no está conectado el telémetro por ultrasonidos trasero y por la tanto se deshabilita el dispositivo y no se pueden obtener medidas a través de él.

A continuación se tiene que seleccionar el sensor que se dispara, la puerta y la secuencia de disparo. Para esta prueba se elige el disparo de ambos sensores (sonar e infrarrojos), la puerta A (en este caso la puerta delantera) y la secuencia de disparo de sensores alternos (operación 3). Se obtienen los siguientes resultados:

Medida 1: Se colocan objetos delante de los sensores 1, 3, 5 y 7, teniendo cuidado para que los objetos puedan ser detectados por ambos sensores.

MEDIDA 1						
	Sensor	L real (cm)	L medida (cm)	Orientación	Error absoluto (cm)	Error (%)
TELÉMETRO POR ULTRASONIDOS	1	180	176,8	-	3,2	1,78
	7	111,4	111,5	-	0,1	0,09
TELÉMETRO POR INFRARROJOS	1	17,3	17,5	-	0,2	1,16
	3	38,3	41,1	-	2,8	7,31
	5	50,4	48,8	-	1,6	3,17
	7	85	+80	-	-	-
BRÚJULA DIGITAL	-	-	-	7,1ºN	-	-

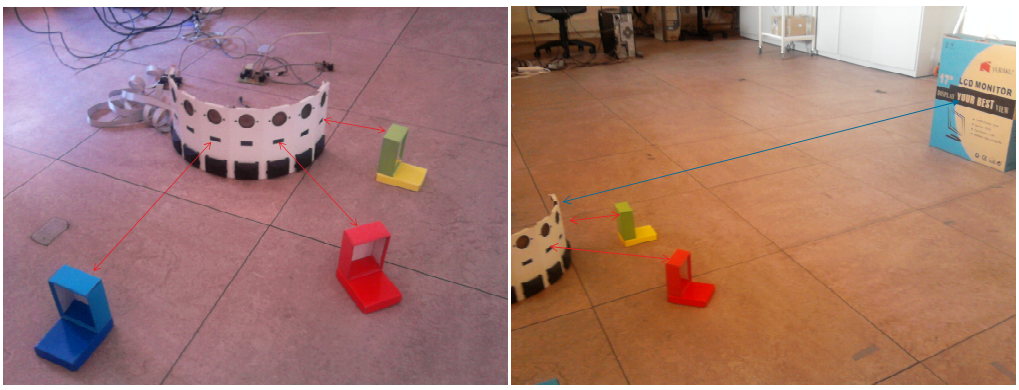


Ilustración 64: Realización de medidas con sonar e IR.

Se puede observar que se obtienen resultados bastante aceptables, por debajo de los 3cm de error. En el caso del telémetro por ultrasonidos se obtiene una mayor precisión que con el telémetro por infrarrojos. Esto se debe a que la exactitud en la medida, con los sensores de ultrasonidos, no depende del objeto contra el que choca la señal, no sucede esto mismo con los sensores de infrarrojos.

Mediante la colocación de sensores de ultrasonidos e infrarrojos alrededor de la base del robot, se ha conseguido un buen sistema de medición de distancias a objetos, puesto que son sensores complementarios. Teniendo en cuenta los resultados experimentales, para medir distancias a objetos en un radio de distancias pequeño (en torno a unos 60cm para evitar el efecto de la reflectividad) se utilizan los sensores de infrarrojos y para medir distancias a objetos en un radio de distancias grande (50cm-10m) se utilizan los sensores por ultrasonidos.

Medida 2: Para comprobar el funcionamiento de la brújula se varía la posición de la placa de control y también se cambian las distancias a las que se encuentran los objetos, obteniendo los siguientes resultados:

MEDIDA 2						
	Sensor	L real (cm)	L medida (cm)	Orientación	Error absoluto (cm)	Error (%)
TELÉMETRO POR ULTRASONIDOS	3	93,6	92,5	-	1,1	1,18
TELÉMETRO POR INFRARROJOS	1	85	+80	-	-	-
	3	27,8	30,5	-	2,7	9,71
	5	32,7	31,8	-	0,9	2,75
	7	16	16,6	-	0,6	3,75
BRÚJULA DIGITAL	-	-	-	45ºN	-	-

Se puede observar como el valor de la orientación ha variado y se comprueba mediante una brújula magnética que es la posición correcta. En cuanto a las distancias obtenidas con los telémetros se vuelve a estar en el margen de menos de 3cm de error y tendiendo más precisión con los ultrasonidos que con los infrarrojos.

Posteriormente se desconecta el telémetro por infrarrojos de la puerta trasera obteniéndose el siguiente mensaje de error mostrado por pantalla:

```
ERROR JUMPERS TELÉMETROS IR Y SONAR PUERTA TRASERA. COLOQUELOS CORRECTAMENTE.
FIN DEL PROGRAMA, VUELVA A INICIAR TODO EL SISTEMA CUANDO SOLUCIONE EL FALLO.
```

Se conecta de nuevo el telémetro por infrarrojos trasero y se reinicia todo el sistema reseteando todas las placas. Seguidamente se conecta el jumper situado en el PIN_C2 del telémetro por infrarrojos de la puerta trasera, por lo tanto, ambos telémetros por infrarrojos tienen la misma asignación de puerta, se muestra el siguiente mensaje de error por pantalla:

```
ERROR JUMPERS TELÉMETROS IR PUERTA DELANTERA Y TRASERA. COLOQUELOS
CORRECTAMENTE. FIN DEL PROGRAMA, VUELVA A INICIAR TODO EL SISTEMA CUANDO
SOLUCIONE EL FALLO.
```

Mediante estas pruebas se ha comprobado el correcto funcionamiento del sistema, tanto en modo normal como en el caso de la existencia de fallos.

4.3.1 Frecuencia en la obtención de medidas.

El tiempo que se tarda en obtener las medidas de los sensores dependerá de la operación que se quiere realizar y del tipo de sensor que se dispare.

Si se tiene en cuenta únicamente el disparo de los sensores infrarrojos, el tiempo en realizar las operaciones 1, 2 y 4 (disparo de los 8 sensores) considerando los tiempos de retardo será:

$$T_{\acute{o}ptimo\ muestreo\ 1\ sensor} = 70ms + (0.1ms \times 2 \times 8) + 1.5ms = \ 73.1\ ms$$

$$T_{\acute{o}ptimo\ muestreo\ 8\ sensores} = 73.1 \times 8 = 584.8\ ms$$

De la misma forma, el tiempo en realizar la operación 3 será:

$$T_{\acute{o}ptimo\ muestreo\ 4\ sensores} = 73.1 \times 4 = 292.4\ ms$$

Si se tiene en cuenta únicamente el disparo de los sensores de ultrasonidos el tiempo en realizar las operaciones 1, 2 y 4 (disparo de los 8 sensores) considerando los tiempos de retardo será:

$$T_{m\acute{a}ximo\ muestreo\ 1\ sensor} = 40\ ms\ [14]$$

$$T_{m\acute{a}ximo\ muestreo\ 8\ sensores} = 40 \times 8 = 320\ ms$$

De la misma forma, el tiempo en realizar la operación 3 será:

$$T_{m\acute{a}ximo\ muestreo\ 4\ sensores} = 40 \times 4 = 160\ ms$$

Si se decide realizar el disparo de ambos sensores se tienen unos tiempos de obtención de medidas, para los operaciones 1, 2 y 4:

$$T = 584.8 + 320 = 904.8\, ms$$

Esto quiere decir que se puede obtener las distancias medidas por todos los sensores de la base del robot en menos de un segundo. De la misma forma el tiempo empleado en obtener las medidas si se realiza la operación 3 será:

$$T = 292.4 + 160 = 452.4\, ms$$

Esto significa que se puede obtener las medidas por los cuatro sensores en menos de medio segundo.

En la fase de inicialización del sistema, el tiempo máximo que se emplea en conocer los dispositivos disponibles para realizar las medidas y la perfecta colocación de los jumpers en cada uno de ellos es:

$$T = 200 \times 4 = 800\, ms$$

Con todos estos datos se puede concluir que el tiempo de muestreo es aceptable, puesto que el robot en un tiempo máximo de un segundo puede tener una visión completa de todos los obstáculos que tiene alrededor suyo.

4.3.2 Restricciones del sistema general.

A pesar de haberse desarrollado un sistema eficaz para el cálculo de distancias a objetos y, como consecuencia, poder desarrollar una habilidad en el robot que le permita desplazarse por su entorno, el sistema tiene algunas restricciones. Se detallan las más importantes a continuación:

- Para el cálculo de distancias mediante los sensores de infrarrojos, existen restricciones en cuanto a la forma, tamaño y superficie del objeto sobre el que la luz infrarroja es reflejada. Por ejemplo, como la luz es proyectada desde el sensor de forma lineal la altura del objeto no puede ser inferior a 15cm y la superficie del mismo debe ser plana.

- En distancias inferiores a los 25cm no se puede asegurar que el objeto detectado esté realmente a la distancia medida, sobre todo si es la primera vez que es detectado, debido a la curva DEC vs L de los sensores infrarrojos.
- Para poder obtener la orientación del robot, la placa de control que integra a la brújula digital se debe colocar en una parte del robot en la que no haya influencia de campos magnéticos, es decir, alejada, por ejemplo, de los motores.
- La frecuencia para obtener las medidas es de una por segundo aproximadamente, si se quiere una mayor frecuencia habría que utilizar otro tipo de sensores.
- Las superficies esquinadas pueden hacer que se obtengan medidas incorrectas tanto si se disparan los sensores infrarrojos como los sensores de ultrasonidos, debido a rebotes en la superficie.

CAPÍTULO 5.- CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se van a exponer las conclusiones que se han obtenido en la realización del proyecto, indicando si se han conseguido obtener o no los objetivos marcados. Después se describirán propuestas de trabajos futuros, a tener en cuenta para realizar en otros proyectos.

5.1 Conclusiones.

A la vista de los resultados experimentales obtenidos, se puede concluir que se han conseguido los objetivos propuestos al inicio de este proyecto.

Se ha diseñado un telémetro mediante sensores de infrarrojos, consiguiendo medir distancias a objetos cercanos al robot, con un error en la medida aceptable en un radio de distancias pequeño.

Se ha implementado el software necesario para obtener la orientación del robot obtenida a través de una brújula digital y para la integración del telémetro por ultrasonidos dentro del sistema.

De igual forma se ha conseguido diseñar un sistema de control sensorial centralizado y monoprocesador, desarrollando el hardware y el software necesarios para integrar el telémetro por infrarrojos, el telémetro por ultrasonidos y una brújula digital, permitiendo una comunicación bidereccional entre los diferentes dispositivos y la placa de control. También se ha desarrollado el software necesario para permitir el control de todo el sistema desde el PC del robot (driver).

Se ha conseguido desarrollar un sistema robusto y modular, que funciona aunque se produzca un fallo en algún dispositivo (salvo excepciones), de esta manera se consigue un fácil manejo, con poco mantenimiento y alta operatividad.

5.2 Trabajos futuros.

Se plantean diferentes desarrollos como trabajos futuros:

5.2.1 Integración de todo el sistema en una sola placa.

Debido a los problemas de espacio que hay en el robot, se propone diseñar una placa en la que se integren los dos telémetros por infrarrojos, los dos telémetros por ultrasonidos y la paca de control. Hay que tener especial cuidado con el magnetismo generado por el transformador del medidor por ultrasonidos y su interacción en el funcionamiento de los componentes, como, por ejemplo, la brújula digital.

5.2.2 Comunicación al PC del robot mediante USB.

Se propone realizar la comunicación entre el PC del robot y la placa de control mediante el protocolo USB. Para ello habría que diseñar una placa de control con el hardware necesario que establece el protocolo e implementar el software para poder establecer la comunicación.

5.2.3 Comunicación del sistema sin cables (wireless).

Para adaptar el sistema a los desarrollos actuales, se propone realizar la comunicación entre los medidores y la placa de control y entre la placa de control y el PC del robot mediante comunicación sin cables, por ejemplo, utilizando la tecnología bluetooth. Como desventaja tiene la menor velocidad de transmisión de datos respecto a la que se puede alcanzar con la comunicación con cables, pero para este proyecto la velocidad de transmisión no es una característica decisiva. Para poder establecer este tipo de comunicación habría que desarrollar el hardware y software necesarios.

5.2.4 Sustituir los sensores de infrarrojos.

Los sensores de infrarrojos que tiene integrados la base del robot están descatalogados, porque son antiguos y no se utilizan en la actualidad. Para una buena mantenibilidad del sistema y mejorar las prestaciones del mismo se recomienda cambiar estos sensores de infrarrojos por otros de mejores características y mayor rango de medida. Actualmente se utilizan sensores como el Sharp GP2D12 o el Sharp IS471F.

ANEXOS

ANEXO A: DATASHEETS.

SENSOR GP2D02

■ Electro-optical Characteristics

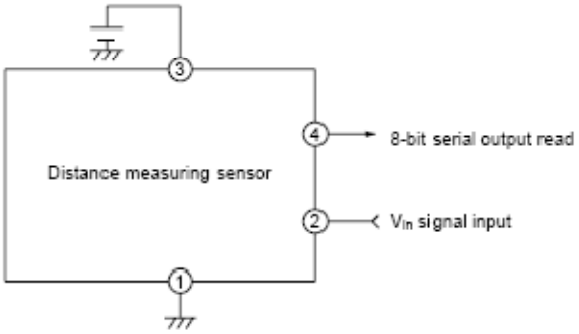
(Ta=25°C,Vcc=5V)

Parameter		Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Distance measuring range		Δ L	*1		10	-	80	cm
Output terminal voltage		V_{OH}	Output voltage at High	L= 20cm	V_{CC} - 0.3	-	-	V
		V_{OL}	Output voltage at Low	*1	-	-	0.3	V
Distance characteristics of output		D	L= 80cm, *1		-	75	-	DEC
		Δ D	Output change at L=80 cm to 20 cm, *1		48	58	68	DEC
Dissipation current	at operating	I_{CC}	L= 20cm, *1, *2		-	22	35	mA
	at OFF-state	I_{off}	L= 20cm, *1		-	3	8	μA
Vin terminal current		I_{vin}	Vin= 0V		-	- 170	- 280	μA

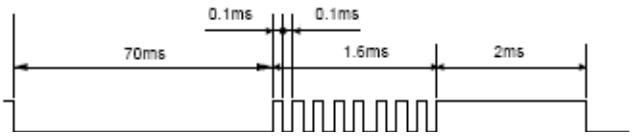
Note) L : Distance to reflective object
DEC : Decimalized value of sensor output (8-bit serial)
*1 Reflective object : White paper (reflectivity : 90%)
*2 Average dissipation current value during distance measuring operation when detecting of input signal, Vin as shown in the timing chart
*3 Vin terminal : Open drain drive input.
Conditions : Vin terminal current at Vin OFF-state : -1 μA
Vin terminal current at Vin ON-state : 0.3V

■ Test Circuit

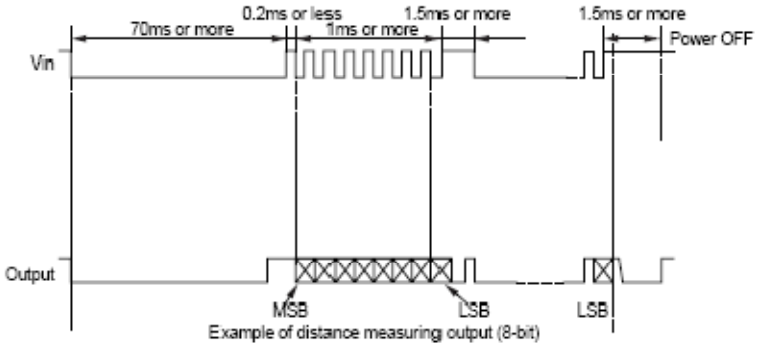
1. Test circuit



2. Vin input signal for measurement



■ Timing Chart



Example of distance measuring output (8-bit)

Fig. 1 Distance Measuring Output vs. Distance to Reflective Object

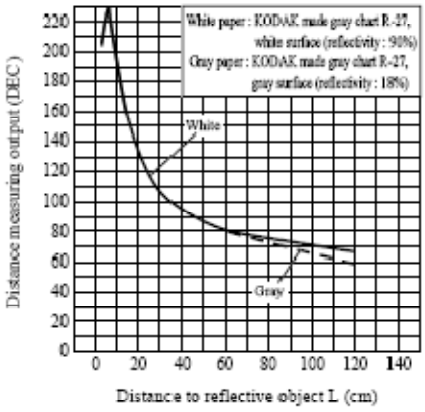
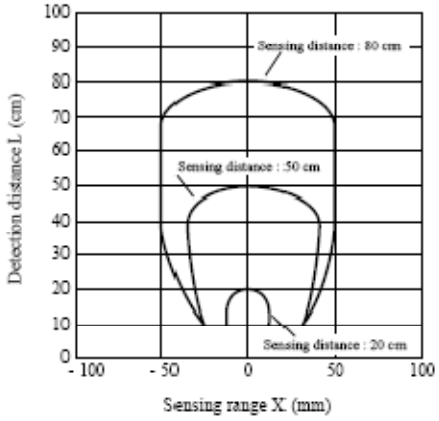


Fig. 2 Detection Distance vs. Sensing Range



Test Method for Sensing Range Characteristics

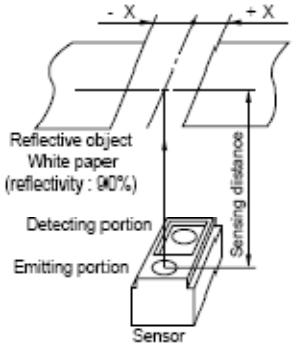
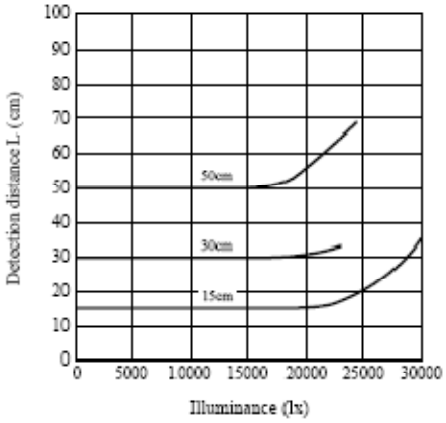
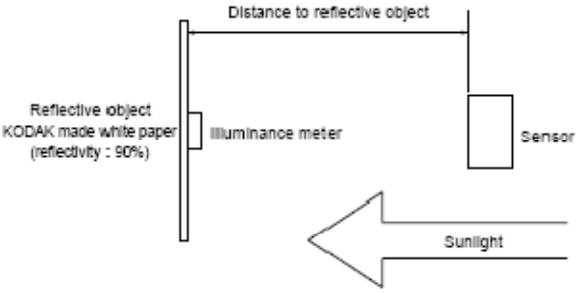


Fig. 3 Detection Distance vs. Illuminance



Test Method for Anti External Disturbing Light Characteristics



DIODO 1N4148

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 60134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{RRM}	repetitive peak reverse voltage		–	100	V
V_R	continuous reverse voltage		–	100	V
I_F	continuous forward current	see Fig.2; note 1	–	200	mA
I_{FRM}	repetitive peak forward current		–	450	mA
I_{FSM}	non-repetitive peak forward current	square wave; T_j = 25 °C prior to surge; see Fig.4			
		t = 1 µs	–	4	A
		t = 1 ms	–	1	A
		t = 1 s	–	0.5	A
P_{tot}	total power dissipation	T_{amb} = 25 °C; note 1	–	500	mW
T_{stg}	storage temperature		–65	+200	°C
T_j	junction temperature		–	200	°C

Note

1. Device mounted on an FR4 printed-circuit board; lead length 10 mm.

ELECTRICAL CHARACTERISTICS

T_j = 25 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_F	forward voltage	see Fig.3			
	1N4148	I_F = 10 mA	–	1	V
	1N4448	I_F = 5 mA	0.62	0.72	V
		I_F = 100 mA	–	1	V
I_R	reverse current	V_R = 20 V; see Fig.5		25	nA
		V_R = 20 V; T_j = 150 °C; see Fig.5	–	50	µA
I_R	reverse current; 1N4448	V_R = 20 V; T_j = 100 °C; see Fig.5	–	3	µA
C_d	diode capacitance	f = 1 MHz; V_R = 0 V; see Fig.6	–	4	pF
t_{rr}	reverse recovery time	when switched from I_F = 10 mA to I_R = 60 mA; R_L = 100 Ω; measured at I_R = 1 mA; see Fig.7	–	4	ns
V_{fr}	forward recovery voltage	when switched from I_F = 50 mA; t_r = 20 ns; see Fig.8	–	2.5	V

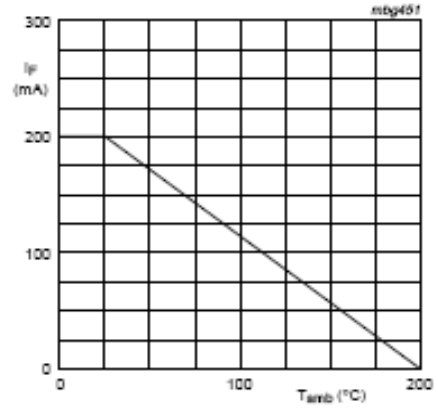
THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th(j-tp)}$	thermal resistance from junction to tie-point	lead length 10 mm	240	K/W
$R_{th(j-a)}$	thermal resistance from junction to ambient	lead length 10 mm; note 1	350	K/W

Note

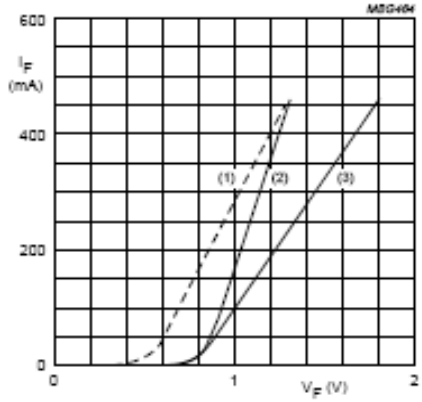
1. Device mounted on a printed-circuit board without metallization pad.

GRAPHICAL DATA



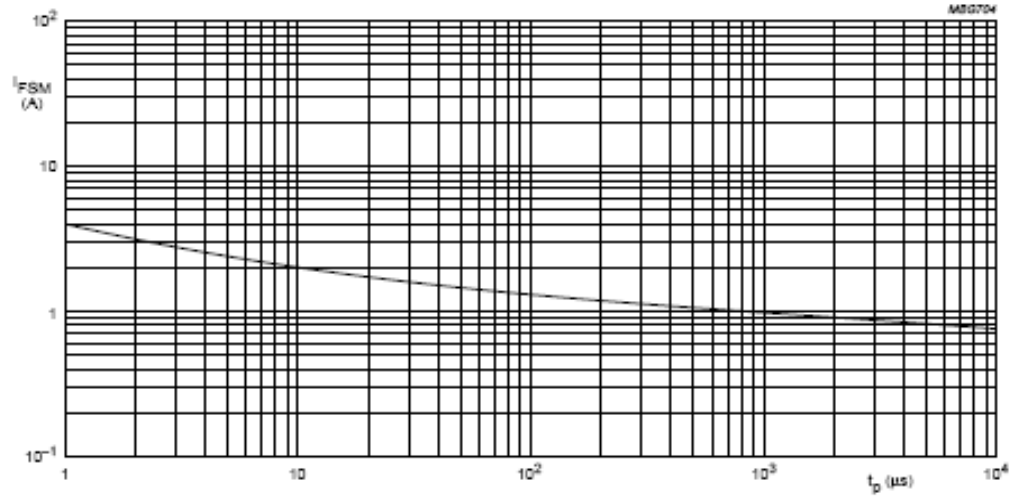
Device mounted on an FR4 printed-circuit board; lead length 10 mm.

Fig.2 Maximum permissible continuous forward current as a function of ambient temperature.



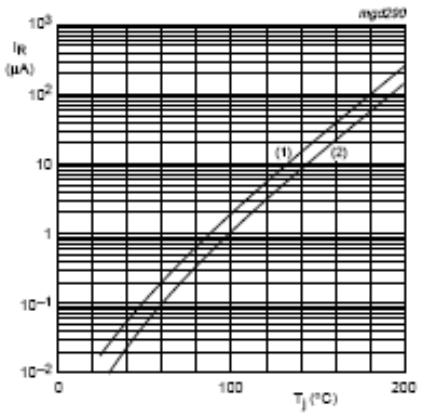
(1) T_j = 175 °C; typical values.
(2) T_j = 25 °C; typical values.
(3) T_j = 25 °C; maximum values.

Fig.3 Forward current as a function of forward voltage.



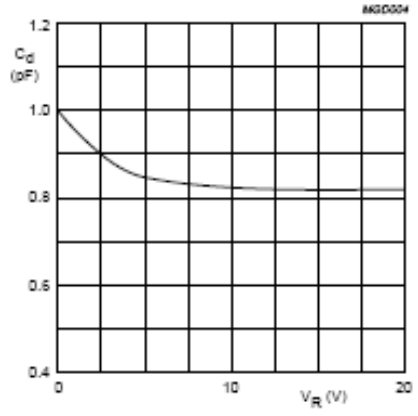
Based on square wave currents.
T_j = 25 °C prior to surge.

Fig.4 Maximum permissible non-repetitive peak forward current as a function of pulse duration.



(1) V_R = 75 V; typical values.
(2) V_R = 20 V; typical values.

Fig.5 Reverse current as a function of junction temperature.

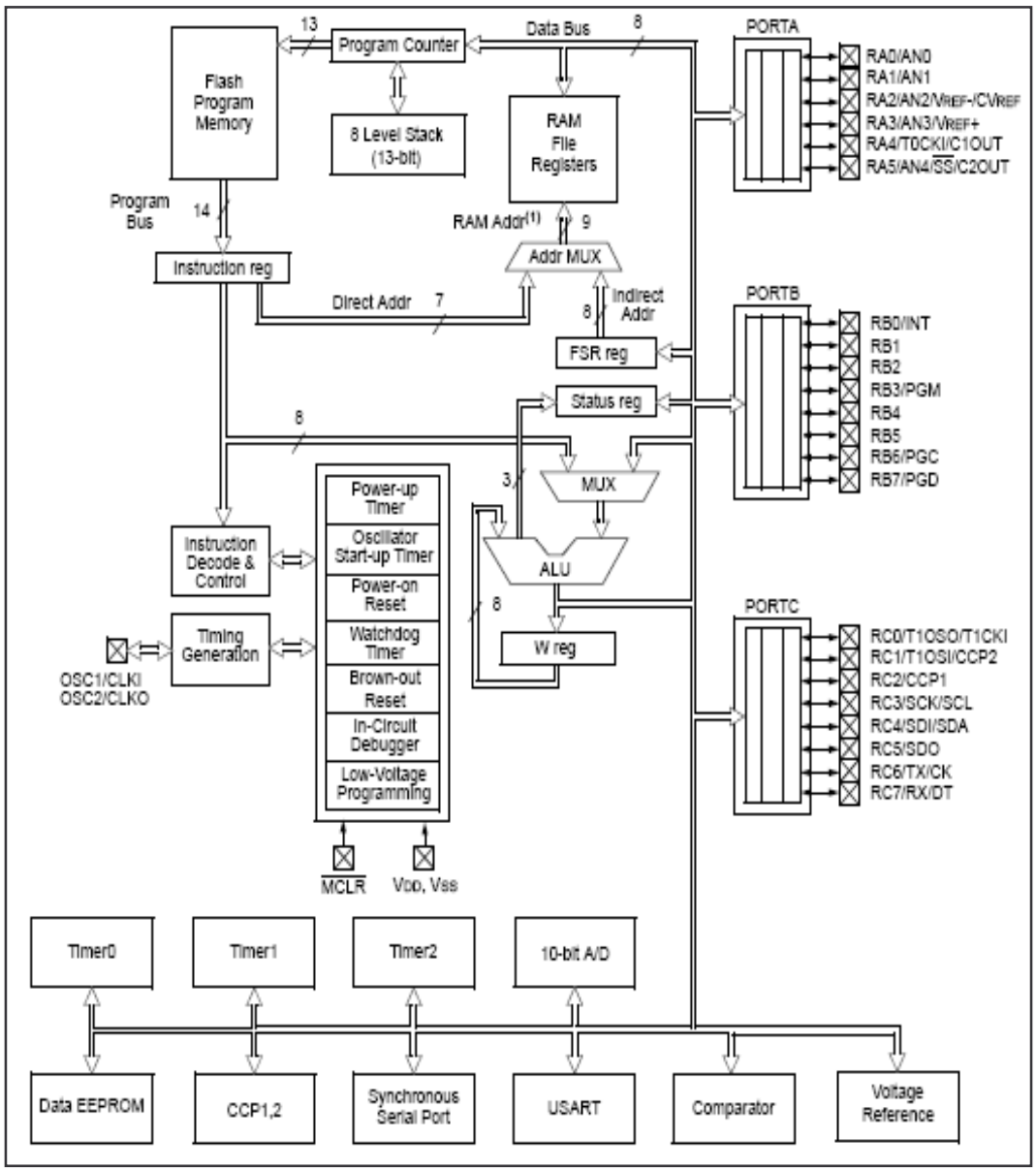


f = 1 MHz; T_j = 25 °C.

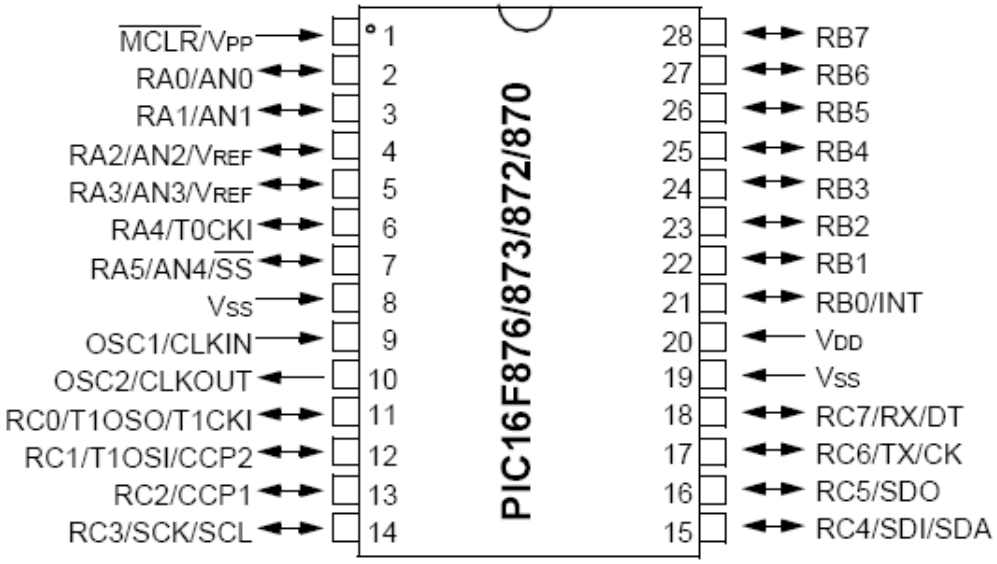
Fig.6 Diode capacitance as a function of reverse voltage; typical values.

PIC 16F876

Diagrama de bloques:



Descripción de los pines:

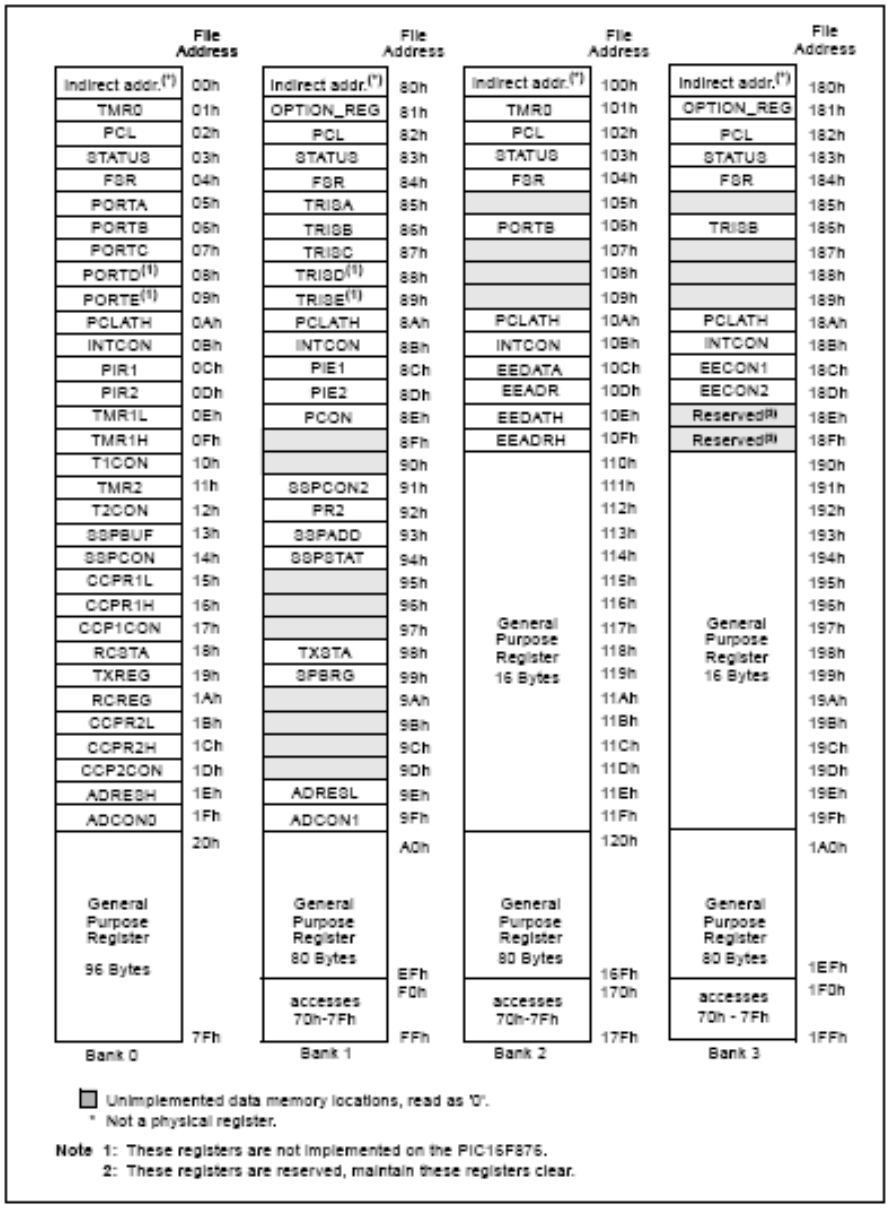


NOMBRE DEL PIN	PIN	TIPO	TIPO DE BUFFER	DESCRIPCIÓN
***MCLR*/Vpp**	1	I/P[1]	ST[2]	Entrada del Master clear (Reset) o entrada de voltaje de programación.
RA0/AN0	2	I/O	TTL	Entrada/salida digital. Entrada analógica 0.
RA1/AN1	3	I/O	TTL	Entrada/salida digital. Entrada analógica 1.
RA2/AN2/Vref-	4	I/O	TTL	Entrada/salida digital. Entrada analógica 2. Referencia negativa de voltaje
RA3/AN3/Vref+	5	I/O	TTL	Entrada/salida digital. Entrada analógica 3. Referencia positiva de voltaje
RA4/T0CKI	6	I/O	ST	Entrada/salida digital (open drain). Entrada de reloj externa del timer0
RA5/AN4/*SS*	7	I/O	TTL	Entrada/salida digital. Entrada analógica 4. Selección de esclavo SPI
Vss	8,19	P	-	Referencia de tierra para los pines lógicos y de I/O
OSC1/CLKIN	9	I	ST/CMOS	Entrada de reloj externa u oscilador de cristal
OSC2/CLKOUT	10	O	-	Salida de reloj externa u oscilador de cristal
RCO/T1OSO/T1CKI	11	I/O	ST	Entrada/salida digital. Salida oscilador timer1. Entrada de reloj externa del timer1
RC1/T1OS1/CCP2	12	I/O	ST	Entrada/salida digital. Entrada oscilador timer1. Entrada Capture2, salida Compare2, salida PWM2.
RC2/CCP1	13	I/O	ST	Entrada/salida digital. Entrada Capture1, salida Compare1, salida PWM1.
RC3/SCK/SCL	14	I/O	ST	Entrada/salida digital. Entrada/salida reloj SPI e I^2C
RC4/SD1/SDA	15	I/O	ST	Entrada/salida digital. Entrada datos SPI. Entrada/salida datos I^2C
RC5/SD0	16	I/O	ST	Entrada/salida digital. Salida datos SPI.
RC6/TX/CK	17	I/O	ST	Entrada/salida digital. Transmisor asíncrono USART. Reloj síncrono USART.
RC7/RX/DT	18	I/O	ST	Entrada/salida digital. Receptor asíncrono USART o datos síncronos USART.
Vdd	20	P	-	Fuente positiva para los pines lógicos y de I/O.
RBO/INT	21	I/O	TTL/ST	Entrada/salida digital. Interrupción externa.
RB1	22	I/O	TTL	Entrada/salida digital.

[1] Power
[2] Schmitt Trigger Input → el disparador de Schmitt permite una mayor inmunidad frente al ruido

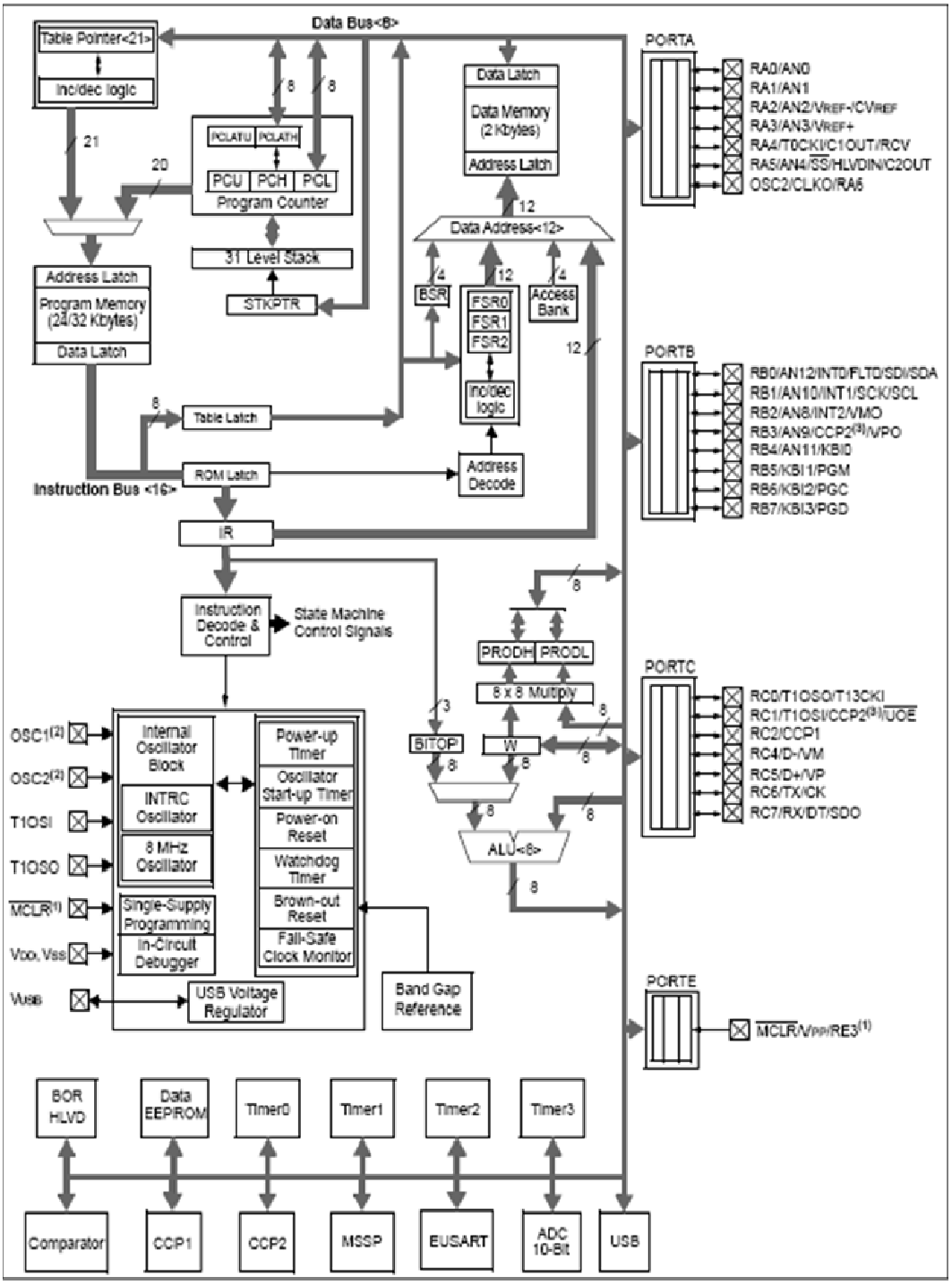
RB2	23	I/O	TTL	Entrada/salida digital.
RB3	24	I/O	TTL	Entrada/salida digital.
RB4	25	I/O	TTL	Entrada/salida digital.
RB5	26	I/O	TTL	Entrada/salida digital.
RB6	27	I/O	TTL	Entrada/salida digital.
RB7	28	I/O	TTL	Entrada/salida digital.

Posiciones de memoria de los registros SFR:

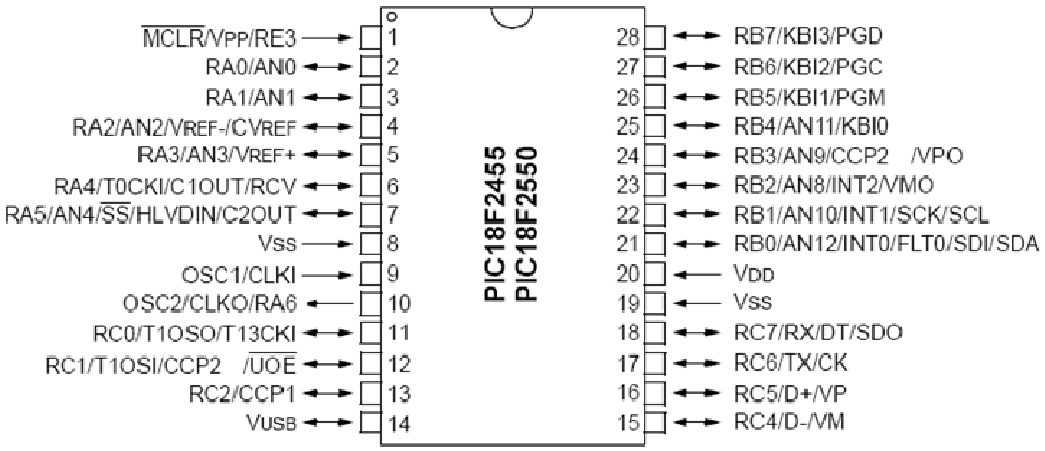


PIC 18F2550

Diagrama de bloques:



Descripción de los pines:



NOMBRE DEL PIN	PIN	TIPO	TIPO DE BUFFER	DESCRIPCIÓN
$\overline{MCLR}$/Vpp/RE3	1	I/P	ST	Entrada del Master clear (Reset). Entrada de voltaje de programación. Entrada digital
RA0/AN0	2	I/O	TTL/Analog	Entrada/salida digital. Entrada analógica 0.
RA1/AN1	3	I/O	TTL/Analog	Entrada/salida digital. Entrada analógica 1.
RA2/AN2/Vref-/CVref	4	I/O	TTL/Analog	Entrada/salida digital. Entrada analógica 2. Referencia negativa de voltaje A/D. Referencia de salida del comparador analógico.
RA3/AN3/Vref+	5	I/O	TTL/Analog	Entrada/salida digital. Entrada analógica 3. Referencia positiva de voltaje A/D.
RA4/T0CKI/C1OUT/RCV	6	I/O	ST/TTL	Entrada/salida digital. Entrada de reloj externa del timer0. Salida comparador 1. USB externo entrada RCV.
RA5/AN4/$\overline{SS}$/HLVDIN/C2OUT	7	I/O	TTL/Analog	Entrada/salida digital. Entrada analógica 4. Selección de esclavo SPI. Detector alto-bajo nivel de tensión de entrada. Salida del comparador 2.
Vss	8,19	P	-	Referencia de tierra para los pines lógicos y de I/O
OSC1/CLKI	9	I	Analog	Entrada de reloj externa u oscilador de cristal
OSC2/CLKO/RA6	10	O	- /TTL	Salida de reloj externa u oscilador de cristal. Entrada/salida digital.
RCO/T1OSO/T1CKI	11	I/O	ST	Entrada/salida digital. Salida oscilador timer1. Entrada de reloj externa del timer1/timer3.
RC1/T1OS1/CCP2/$\overline{UOE}$	12	I/O	ST/CMOS	Entrada/salida digital. Entrada oscilador timer1. Entrada Capture2, salida Compare2, salida PWM2. USB externo entrada OE.
RC2/CCP1	13	I/O	ST	Entrada/salida digital. Entrada Capture1, salida Compare1, salida PWM1.
Vusb	14	O	-	Regulador de tensión interna USB 3.3V
RC4/D-/VM	15	I/O	TTL	Entrada/salida digital. Entrada/salida negativa diferencial USB.USB externo entrada VM
RC5/D+/VP	16	I/O	TTL	Entrada/salida digital. Entrada/salida

				positiva diferencial USB. USB externo entrada VP
RC6/TX/CK	17	I/O	ST	Entrada/salida digital. Transmisor asíncrono EUSART. Reloj síncrono EUSART
RC7/RX/DT/SDO	18	I/O	ST	Entrada/salida digital. Receptor asíncrono EUSART o datos síncronos EUSART. Salida datos SPI.
Vdd	20	P	-	Fuente positiva para los pines lógicos y de I/O.
RB0/AN12/INT0/FLT0/SDI/SDA	21	I/O	TTL/ST/Analog	Entrada/salida digital. Entrada analógica 12. Interrupción externa 0. Entrada PWM. Entrada de datos SPI. Entrada/salida de datos I^2C.
RB1/AN10/INT1/SCK/SCL	22	I/O	TTL/ST/Analog	Entrada/salida digital. Entrada analógica 10. Interrupción externa 1. Entrada/salida reloj SPI e I^2C
RB2/AN8/INT2/VMO	23	I/O	TTL/ST/Analog	Entrada/salida digital. Entrada analógica 8. Interrupción externa 2. USB externo salida VMO.
RB3/AN9/CCP2/VPO	24	I/O	TTL/ST/Analog	Entrada/salida digital. Entrada analógica 9. Entrada Capture2, salida Compare2, salida PWM2. USB externo salida VPO.
RB4/AN11/KBI0	25	I/O	TTL/Analog	Entrada/salida digital. Entrada analógica 11. Pin interrupt-on-change
RB5/KBI1/PGM	26	I/O	TTL/ST	Entrada/salida digital. Pin interrupt-on-change. Pin bajo voltaje programación ICSP.
RB6/KBI2/PGC	27	I/O	TTL/ST	Entrada/salida digital. Pin interrupt-on-change. Pin de reloj en circuito debugger y programación ICSP
RB7/KBI3/PGD	28	I/O	TTL/ST	Entrada/salida digital. Pin interrupt-on-change. Pin de datos en circuito debugger y programación ICSP.

Posiciones de memoria de los registros SFR:

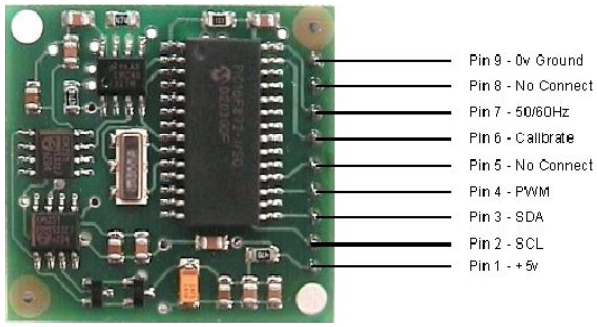
Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(1)	FBFh	CCPR1H	F9Fh	IPR1	F7Fh	UEP15
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	CCPR1L	F9Eh	PIR1	F7Eh	UEP14
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCP1CON	F9Dh	PIE1	F7Dh	UEP13
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR2H	F9Ch	—(2)	F7Ch	UEP12
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR2L	F9Bh	OSCTUNE	F7Bh	UEP11
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—(2)	F7Ah	UEP10
FF9h	PCL	FD9h	FSR2L	FB9h	—(2)	F99h	—(2)	F79h	UEP9
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	—(2)	F78h	UEP8
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	ECCP1DEL	F97h	—(2)	F77h	UEP7
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS	F96h	TRISE(3)	F76h	UEP6
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD(3)	F75h	UEP5
FF4h	PRODH	FD4h	—(2)	FB4h	CMCON	F94h	TRISC	F74h	UEP4
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB	F73h	UEP3
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA	F72h	UEP2
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—(2)	F71h	UEP1
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	—(2)	F70h	UEP0
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—(2)	F6Fh	UCFG
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—(2)	F6Eh	UADDR
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(3)	F6Dh	UCON
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(3)	F6Ch	USTAT
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC	F6Bh	UEIE
FEAh	FSR0H	FCAh	T2CON	FAAh	—(2)	F8Ah	LATB	F6Ah	UEIR
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA	F69h	UIE
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—(2)	F68h	UIR
FE7h	INDF1(1)	FC7h	SSPSTAT	FA7h	EECON2(1)	F87h	—(2)	F67h	UFRMH
FE6h	POSTINC1(1)	FC6h	SSPCON1	FA6h	EECON1	F86h	—(2)	F66h	UFRML
FE5h	POSTDEC1(1)	FC5h	SSPCON2	FA5h	—(2)	F85h	—(2)	F65h	SPPCON(3)
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	—(2)	F84h	PORTE	F64h	SPPEPS(3)
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	—(2)	F83h	PORTD(3)	F63h	SPPCFG(3)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC	F62h	SPPDATA(3)
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB	F61h	—(2)
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA	F60h	—(2)

Note 1: Not a physical register.
2: Unimplemented registers are read as '0'.
3: These registers are implemented only on 40/44-pin devices.

COMPASS MODULE CMPS03

This compass module has been specifically designed for use in robots as an aid to navigation. The aim was to produce a unique number to represent the direction the robot is facing. The compass uses the Philips KMZ51 magnetic field sensor, which is sensitive enough to detect the Earths magnetic field. The output from two of them mounted at right angles to each other is used to compute the direction of the horizontal component of the Earths magnetic field.

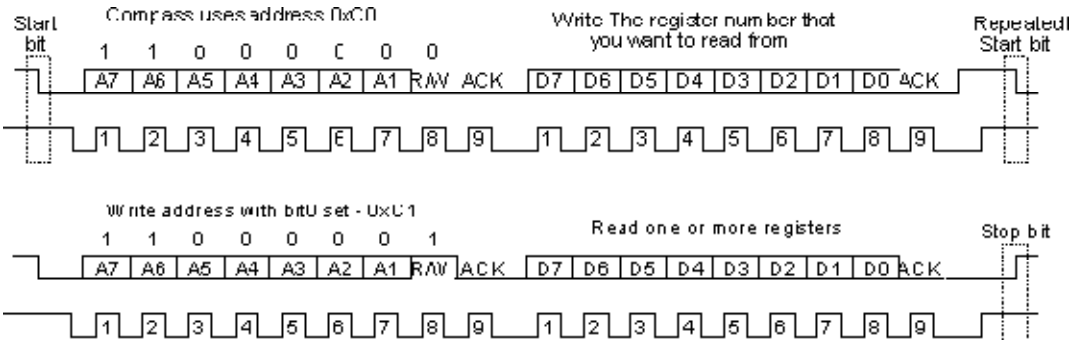
Connections to the compass module:



The compass module requires a 5v power supply at a nominal 15mA.

There are two ways of getting the bearing from the module. A PWM signal is available on pin 4, or an I2C interface is provided on pins 2, 3. The PWM signal is a pulse width modulated signal with the positive width of the pulse representing the angle. The pulse width varies from 1mS (0°) to 36.99mS (359.9°) – in other words 100uS/° with a +1mS offset. The signal goes low for 65mS between pulses, so the cycle time is 65mS + the pulse width - ie. 66ms-102ms.The pulse is generated by a 16 bit timer in the processor giving a 1uS resolution, however I would not recommend measuring this to anything better than 0.1° (10uS). Make sure you connect the I2C pins, SCL and SDA, to the 5v supply if you are using the PWM, as there are no pull-up resistors on these pins.

Pin 2, 3 are an I^2C interface and can be used to get a direct readout of the bearing. If the I2C interface is not used then these pins should be pulled high (to +5v) via a couple of resistors. Around 47k is ok, the values are not at all critical.



I^2C communication protocol with the compass module is the same as popular eeprom's such as the 24C04. First send a start bit, the module address (0XC0) with the read/write bit low, then the register number you wish to read. This is followed by a repeated start and the module address again with the read/write bit high (0XC1). You now read one or two bytes for 8bit or 16bit registers respectively. 16bit registers are read high byte first. The compass has a 16 byte array of registers, some of which double up as 16 bit registers as follows.

Register	**Function**
0	Software Revision Number
1	Compass Bearing as a byte, i.e. 0-255 for a full circle
2, 3	Compass Bearing as a word, i.e. 0-3599 for a full circle, representing 0-359.9 degrees.
4, 5	Internal Test - Sensor1 difference signal - 16 bit signed word
6,7	Internal Test - Sensor2 difference signal - 16 bit signed word
8, 9	Internal Test - Calibration value 1 - 16 bit signed word
10, 11	Internal Test - Calibration value 2 - 16 bit signed word
12	Unused - Read as Zero
13	Unused - Read as Zero
14	Unused - Read as Undefined
15	Calibrate Command - Write 255 to perform calibration step. See text.

Register 0 is the Software revision number (8 at the time of writing). Register 1 is the bearing converted to a 0-255 value. This may be easier for some applications than 0-360 which requires two bytes. For those who require better resolution registers 2 and 3 (high byte first) are a 16 bit unsigned integer in the range 0-3599. This represents 0-359.9°. Registers 4 to 11 are internal test registers and 12, 13 are unused. Register 14 is undefined. Don't read them if you don't want them - you'll just waste your I^2C bandwidth. Register 15 is used to calibrate the compass.

The I^2C interface does not have any pull-up resistors on the board, these should be provided elsewhere, most probably with the bus master. They are required on both the SCL and SDA lines, but only once for the whole bus, not on each module. I suggest a value of 1k8 if you are going to be working up to 400KHz and 1k2 or even 1k if you are going up to 1MHz. The compass is designed to work at up to the standard clock speed (SCL) of 100KHz, however the clock speed can be raised to 1MHZ providing the following precaution is taken. At speeds above around 160KHz the CPU cannot respond fast enough to read the I2C data. Therefore a small delay of 50uS should be inserted either side of writing the register address. No delays are required anywhere else in the sequence. By doing this, I have tested the compass module up to 1.3MHz SCL clock speed.

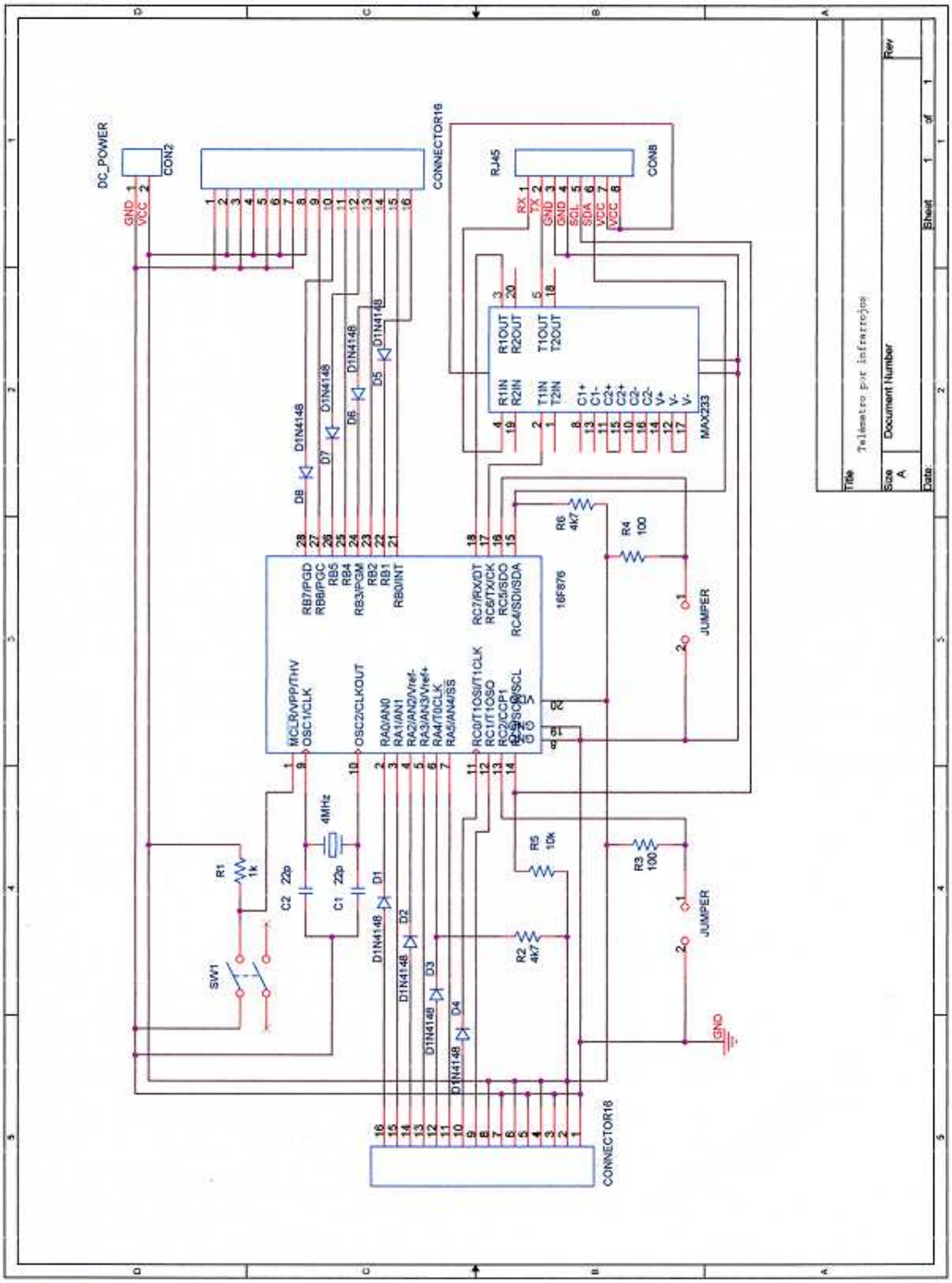
Pin 7 is an input pin selecting either 50Hz (low) or 60Hz (high) operation. I added this option after noticing a jitter of around 1.5° in the output. The cause was the 50Hz mains field in my workshop. By converting in synchronism with the mains frequency this was reduced to around 0.2°. An internal conversion is done every 40mS (50Hz) or every 33.3mS (60Hz). The pin has an on-board pull-up can be left unconnected for 60Hz operation. There is no synchronism between the PWM or I2C outputs and the conversion. They both retrieve the most recent internal reading, which is continuously converted, whether it is used or not.

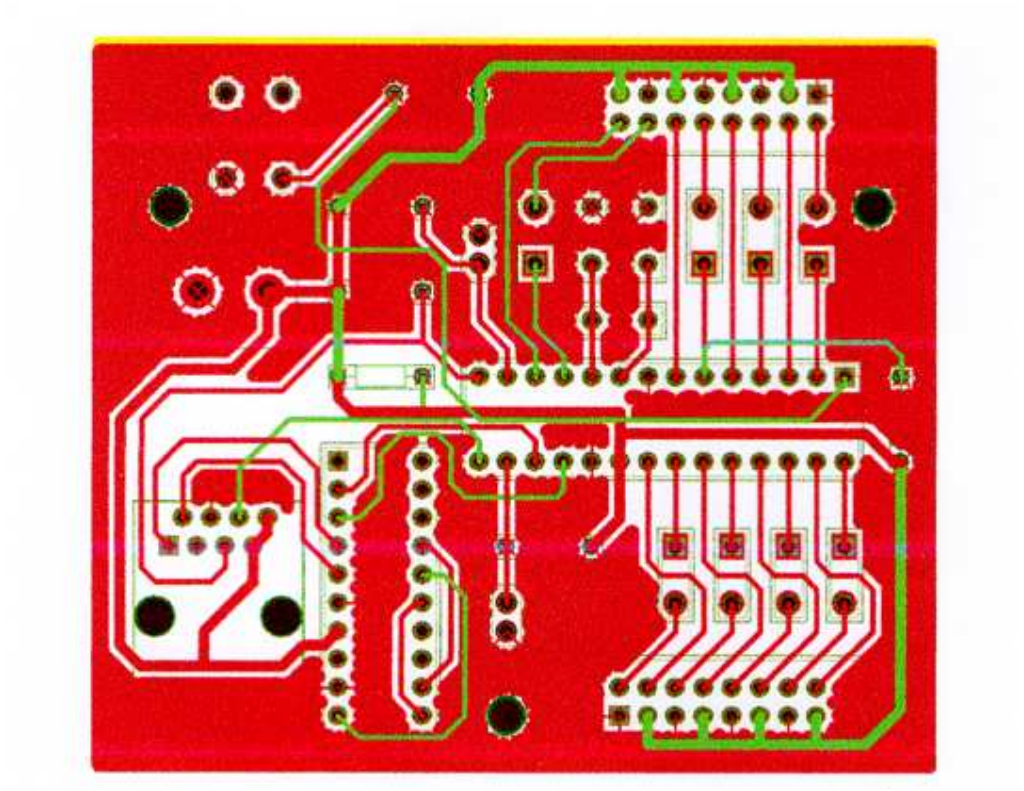
Pin 6 is used to calibrate the compass. The calibrate input (pin 6) has an on-board pull-up resistor and can be left unconnected after calibration.

Pins 5 and 8 are No Connect. Actually pin 8 is the processor reset line and has an on-board pull-up resistor. It is there so that we can program the processor chip after placement on the PCB.

ANEXO B: ESQUEMÁTICO Y LAYOUT

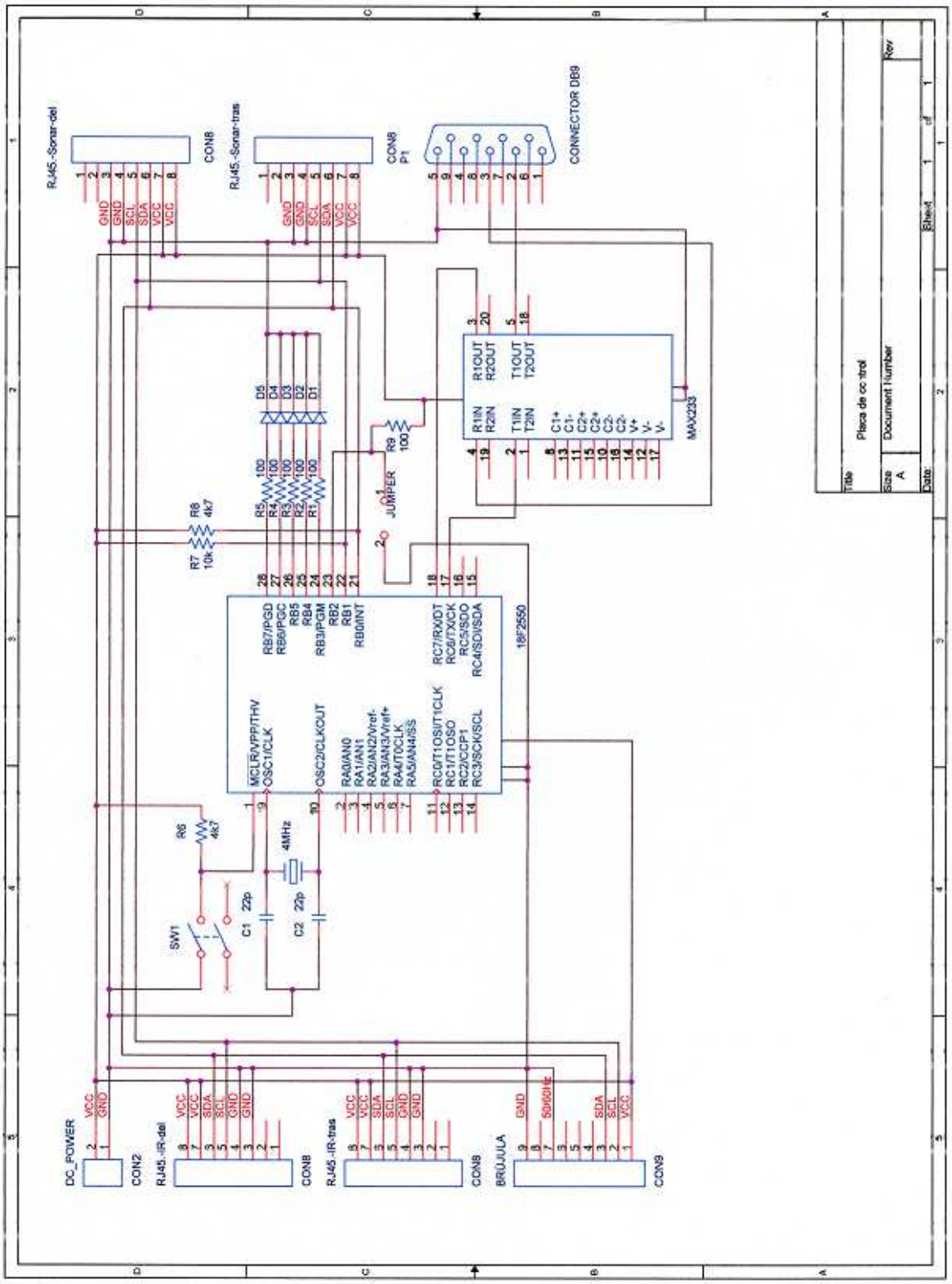
TELÉMETRO POR INFRARROJOS

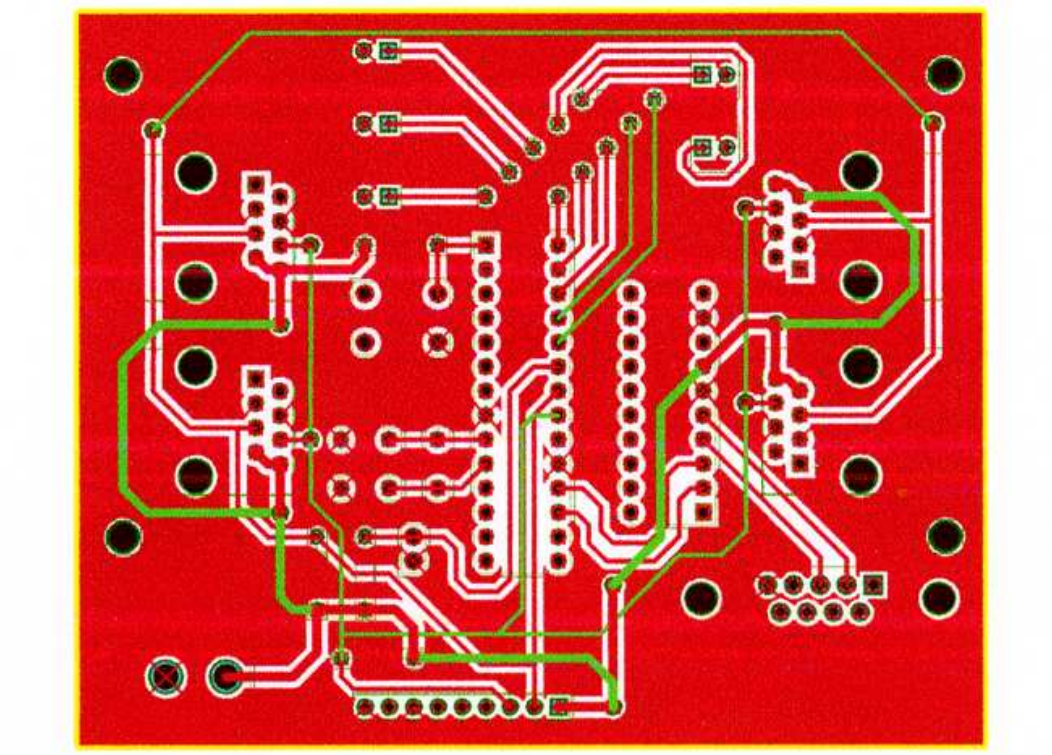

DC_POWER
CON2
GND
VCC
CONNECTOR16
RJ45
CON8
RX
TX
GND
SCL
SDA
VCC
MAX233
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
T1IN
T2IN
C1+
C1-
C2+
C2-
V+
V-
D1N4148
D1
D2
D3
D4
D5
D6
D7
D8
16F876
MCLR/VPP/THV
OSC1/CLK
OSC2/CLKOUT
RB7/PGD
RB6/PGC
RB5
RB4
RB3/PGM
RB2
RB1
RB0/INT
RC7/RX/DT
RC6/TX/CK
RC5/SDO
RC4/SDI/SDA
RA0/AN0
RA1/AN1
RA2/AN2/Vref-
RA3/AN3/Vref+
RA4/T0CLK
RA5/AN4/SS
RC0/T1OSO/T1CLK
RC1/T1OSI
RC2/CCP1
R1
1k
R2
4k7
R3
100
R4
100
R5
10k
R6
4k7
C1
22p
C2
22p
4MHz
SW1
JUMPER
JUMPER
GND
CONNECTOR16
Telemetro por infrarrojos
Title
Size
A
Document Number
Rev
Date:
Sheet
1
of
1



El rutado de las pistas se ha realizado en la capa *top* (pistas en color verde) y en la capa *bottom* (pistas en color rojo). El plano de masa se realiza en la capa *bottom*. Se han realizado tres agujeros en la placa para poder integrarla a la estructura de la base del robot.

PLACA DE CONTROL


RJ45.-Sonar-del
CON8
RJ45.-Sonar-tras
CON8
P1
CONNECTOR DB9
GND
GND
SCL
SDA
VCC
VCC
D5
D4
D3
D2
D1
R5 100
R4 100
R3 100
R2 100
R1 100
R9 100
JUMPER
R8 4k7
R7 10k
R6 4k7
MAX233
R1IN
R2IN
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
C1+
C1-
C2+
C2-
V+
V-
16F2550
RB7/PGD
RB6/PGC
RB5
RB4
RB3/PGM
RB2
RB1
RB0/INT
RC7/RX/DT
RC6/TX/CK
RC5/SDO
RC4/SDI/SDA
MCLR/VPP/THV
OSC1/CLK
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/Vref-
RA3/AN3/Vref+
RA4/T0CLK
RA5/AN4/SS
RC0/T1OSI/T1CLK
RC1/T1OSO
RC2/CCP1
RC3/SCK/SCL
4MHz
C1 22p
C2 22p
SW1
Title
Placa de control
Size
A
Document Number
Rev
Date:
Sheet
DC_POWER
CON2
RJ45.-IR-del
CON8
RJ45.-IR-tras
CON8
BRÚJULA
CON9
50/60Hz



El rutado de las pistas se ha realizado en la capa *top* (pistas en color verde) y en la capa *bottom* (pistas en color rojo). El plano de masa se realiza en la capa *bottom*. Se han realizado cuatro agujeros en la placa para poder integrarla a la estructura del robot.

ANEXO C: CÓDIGO

TELÉMETRO POR INFRARROJOS PUERTA DELANTERA

```
//PROGRAMA TELÉMETRO POR INFRARROJOS PUERTA DELANTERA

#include <16F876.h>

#fuses XT, NOLVP, NOWRT, NOBROWNOUT, NOCPD, NOWDT, NOPROTECT
#use delay(clock=4000000)     // Velocidad del oscilador : 4 Mhz
#use RS232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7) //Se definen parámetros para
comunicación RS-232
#use i2c (slave, sda=PIN_C4, scl=PIN_C3, address=0x10, FAST) //Se definen
parámetros para comunicación I2C

#bit R_W = 0x94.2
#bit SSPIF = 0x0C.3
#bit T0IF = 0x0B.2
#byte TMR0 = 0x01
#byte OPTION = 0x81
#byte ADCON1 = 0x9F

BYTE operacion;
BYTE medida[10];
BYTE estado[3];
BYTE CRC;
int flag=0;
int k;
int count=0;

void realizar_operacion(void);
unsigned char GP2D02(int sensor);
void tiempo_pulso(void);
void tiempo_espera(void);
void medida_realizada(void);
void calculo_CRC(void);
void com_i2c(void);
void com_232(void);
float convertir_valor(BYTE medida);

#INT_SSP
void ssp_interrupt()
{
   if(i2c_poll())
   {
      operacion=i2c_read();
      count++;
   }
   if(R_W==1) //La placa de control quiere recibir datos
   {
      flag=1;
   }
}


void main()
{
   short int comunicacion;
   comunicacion=input(PIN_C2); //Se comprueba el estado del PIN_C2 para saber
si la comunicación se establece con el PC o con la placa de control
   if(comunicacion==0)
   {
      com_i2c(); //La comunicación se establece con la placa de control
   }
   if(comunicacion==1)
   {
      com_232(); //La comunicación se establece con el PC
   }
}
```

```
//Función que se ejecuta cuando se establece la comunicación con la placa de
control
void com_i2c(void)
{
   enable_interrupts(GLOBAL); //Se habilitan todas las interrupciones
   enable_interrupts(INT_SSP); //Se habilita la interrupción módulo MSSP
   while (TRUE)
   {
      while(count!=2); //Se espera a recibir la operación a realizar
      count=0;
      realizar_operacion();
      while(flag==0); //Se espera a que el maestro quiera recibir los datos
      if(operacion==0x05)
      {
         for(k=0;k<=2;k++)
         {
            delay_us(50);
            i2c_write(estado[k]);
         }
      }
      else
      {
         for(k=0;k<=9;k++)
         {
            delay_us(50);
            i2c_write(medida[k]);
         }
      }
      SSPIF=0;
      flag=0;
   }
}

//Función que se ejecuta cuando se establece la comunicación con la placa de
control
void com_232(void)
{
   char seleccion;
   int sensor,auxiliar,i;
   float longitud;

   OPTION = 0b00000101; // Se configura el pre-escaler del timer 0 a 64 y R
pull-up puerto B
   SETUP_ADC_PORTS(NO_ANALOGS);//Puerta A entradas/salidas digitales
   set_tris_a(0x2A); //Se configuran los pines del puerto A como entradas o
salidas
   set_tris_b(0x55); //Se configuran los pines del puerto B como entradas o
salidas
   set_tris_c(0xBE); //Se configuran los pines del puerto C como entradas o
salidas

   output_high(PIN_A0);
   output_high(PIN_A2);
   output_high(PIN_A4);
   output_high(PIN_C0);
   output_high(PIN_B1);
   output_high(PIN_B3);
   output_high(PIN_B5);
   output_high(PIN_B7);

   while(TRUE)
   {
   for(i=0;i<=9;i++)
   {
      medida[i]=0;
   }
   auxiliar=1;
```

```
      do
      {
         printf("\r\nSeleccion de la secuencia de disparo de los sensores de
infrarrojos:\n\n");
         printf("\rPulse la opcion deseada:\n\n");
         printf("\r  1.- Disparos consecutivos de izquierda a derecha\n");
         printf("\r  2.- Disparos consecutivos de derecha a izquierda\n");
         printf("\r  3.- Disparos alternos\n");
         printf("\r  4.- Disparos cada cuatro sensores, uno por
cuadrante\n\n");
         seleccion=getchar();
      }while(seleccion < '1' || seleccion > '4');

   switch(seleccion)
   {
         case '1':
               medida[0]=8;
               for(sensor=1;sensor<=8;sensor++)
               {
                  medida[auxiliar]=GP2D02(sensor);
                  auxiliar++;
               }
               for(auxiliar=1;auxiliar<=8;auxiliar++)
               {
                  if(medida[auxiliar]<=80)
                  {
                    printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",auxiliar);
                  }
                  else
                  {
                     longitud=convertir_valor(medida[auxiliar]);
                     printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,auxiliar);
                  }
               }
               break;
         case '2':
               medida[0]=8;
               for(sensor=8;sensor>=1;sensor--)
               {
                  medida[auxiliar]=GP2D02(sensor);
                  auxiliar++;
               }
               for(auxiliar=1;auxiliar<=8;auxiliar++)
               {
                  if(medida[auxiliar]<=80)
                  {
                    printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",9-auxiliar);
                  }
                  else
                  {
                     longitud=convertir_valor(medida[auxiliar]);
                     printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,9-auxiliar);
                  }
               }
               break;
         case '3':
               medida[0]=4;
               for(sensor=1;sensor<=8;sensor++)
               {
                  if((sensor==1)||(sensor==3)||(sensor==5)||(sensor==7))
                  {
                     medida[auxiliar]=GP2D02(sensor);
                     auxiliar++;
```

```
                }
            }
            for(auxiliar=1;auxiliar<=4;auxiliar++)
            {
                if(medida[auxiliar]<=80)
                {
                  printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(2*auxiliar)-1);
                }
                else
                {
                    longitud=convertir_valor(medida[auxiliar]);
                    printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(2*auxiliar)-1);
                }
            }
            break;
        case '4':
            medida[0]=8;
            for(sensor=1;sensor<=8;sensor++)
            {
                if((sensor==1)||(sensor==5))
                {
                    medida[auxiliar]=GP2D02(sensor);
                    auxiliar++;
                }
            }
            auxiliar=3;
            for(sensor=1;sensor<=8;sensor++)
            {
                if((sensor==2)||(sensor==6))
                {
                    medida[auxiliar]=GP2D02(sensor);
                    auxiliar++;
                }
            }
            auxiliar=5;
            for(sensor=1;sensor<=8;sensor++)
            {
                if((sensor==3)||(sensor==7))
                {
                    medida[auxiliar]=GP2D02(sensor);
                    auxiliar++;
                }
            }
            auxiliar=7;
            for(sensor=1;sensor<=8;sensor++)
            {
                if((sensor==4)||(sensor==8))
                {
                    medida[auxiliar]=GP2D02(sensor);
                    auxiliar++;
                }
            }
            for(auxiliar=1;auxiliar<=2;auxiliar++)
            {
                if(medida[auxiliar]<=80)
                {
                  printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(4*(auxiliar-1))+1);
                }
                else
                {
                    longitud=convertir_valor(medida[auxiliar]);
                    printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(4*(auxiliar-1))+1);
                }
```

```
                }
                for(auxiliar=1;auxiliar<=2;auxiliar++)
                {
                   if(medida[auxiliar+2]<=80)
                   {
                     printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(4*(auxiliar-1))+2);

                   }
                   else
                   {
                      longitud=convertir_valor(medida[auxiliar+2]);
                      printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(4*(auxiliar-1))+2);
                   }
                }
                for(auxiliar=1;auxiliar<=2;auxiliar++)
                {
                   if(medida[auxiliar+4]<=80)
                   {
                     printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(4*(auxiliar-1))+3);
                   }
                   else
                   {
                      longitud=convertir_valor(medida[auxiliar+4]);
                      printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(4*(auxiliar-1))+3);
                   }
                }
                for(auxiliar=1;auxiliar<=2;auxiliar++)
                {
                   if(medida[auxiliar+6]<=80)
                   {
                     printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(4*(auxiliar-1))+4);
                   }
                   else
                   {
                      longitud=convertir_valor(medida[auxiliar+6]);
                      printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(4*(auxiliar-1))+4);
                   }
                }
                break;
             default:
                break;
          }
          delay_ms(10);
      }
}


//Función que dispara los sensores y que se ejecuta cuando se establece
comunicación con la placa de control
void realizar_operacion()
{
   short int puerta;
   int sensor, auxiliar=1;
   OPTION = 0b00000101;
   SETUP_ADC_PORTS(NO_ANALOGS);
   DISABLE_INTERRUPTS(INT_EXT); //Se deshabilita la interrupción externa del
pin RB0
   set_tris_a(0x2A);
   set_tris_b(0x55);
   set_tris_c(0xBE);
```

```
output_high(PIN_A0);
output_high(PIN_A2);
output_high(PIN_A4);
output_high(PIN_C0);
output_high(PIN_B1);
output_high(PIN_B3);
output_high(PIN_B5);
output_high(PIN_B7);

if(operacion==0x01)
{
      for(sensor=1;sensor<=8;sensor++)
      {
           medida[auxiliar]=GP2D02(sensor);
           auxiliar++;
      }
      medida[0]=8;
      calculo_CRC();
      medida[9]=CRC;
}
if(operacion==0x02)
{
      for(sensor=8;sensor>=1;sensor--)
      {
           medida[auxiliar]=GP2D02(sensor);
           auxiliar++;
      }
      medida[0]=8;
      calculo_CRC();
      medida[9]=CRC;
}
if(operacion==0x03)
{
    for(sensor=1;sensor<=8;sensor++)
    {
        if((sensor==1)||(sensor==3)||(sensor==5)||(sensor==7))
        {
             medida[auxiliar]=GP2D02(sensor);
             auxiliar++;
        }
    }
    medida[0]=4;
    calculo_CRC();
    medida[5]=CRC;
}
if(operacion==0x04)
{
    for(sensor=1;sensor<=8;sensor++)
    {
      if((sensor==1)||(sensor==5)){
          medida[auxiliar]=GP2D02(sensor);
          auxiliar++;
      }
    }
    auxiliar=3;
    for(sensor=1;sensor<=8;sensor++)
    {
      if((sensor==2)||(sensor==6)){
          medida[auxiliar]=GP2D02(sensor);
          auxiliar++;
      }
    }
    auxiliar=5;
    for(sensor=1;sensor<=8;sensor++)
    {
      if((sensor==3)||(sensor==7)){
         medida[auxiliar]=GP2D02(sensor);
```

```
            auxiliar++;
          }
        }
        auxiliar=7;
        for(sensor=1;sensor<=8;sensor++)
        {
          if((sensor==4)||(sensor==8)){
            medida[auxiliar]=GP2D02(sensor);
            auxiliar++;
          }
        }
        medida[0]=8;
        calculo_CRC();
        medida[9]=CRC;
    }
    if(operacion==0x05)
    {
        puerta=input(PIN_C5);
        estado[0]=1;
        estado[1]=puerta;
        estado[2]=estado[1];
    }
}


//Función que genera las señales que posibilitan el disparo de los sensores y
recoge los resultados obtenidos
unsigned char GP2D02(int sensor)
{
   unsigned char resultado;
   int i;
   short int estado;

   resultado=0;

   if(sensor==1){
         output_high(PIN_A0);
         tiempo_pulso();
         output_low(PIN_A0);
         medida_realizada();
         for(i=0;i<=7;i++)
         {
            output_high(PIN_A0);
            tiempo_pulso();
            output_low(PIN_A0);
            tiempo_pulso();
            estado=input(PIN_A1);
            resultado=resultado<<1;
            if(estado)bit_set(resultado,0);
         }
         output_high(PIN_A0);
         tiempo_espera();
   }
   else{
      if(sensor==2){
         output_high(PIN_A2);
         tiempo_pulso();
         output_low(PIN_A2);
         medida_realizada();
         for(i=0;i<=7;i++)
         {
            output_high(PIN_A2);
            tiempo_pulso();
            output_low(PIN_A2);
            tiempo_pulso();
            estado=input(PIN_A3);
            resultado=resultado<<1;
```

```
            if(estado)bit_set(resultado,0);
         }
         output_high(PIN_A2);
         tiempo_espera();
      }
      else{
         if(sensor==3){
            output_high(PIN_A4);
            tiempo_pulso();
            output_low(PIN_A4);
            medida_realizada();
            for(i=0;i<=7;i++)
            {
               output_high(PIN_A4);
               tiempo_pulso();
               output_low(PIN_A4);
               tiempo_pulso();
               estado=input(PIN_A5);
               resultado=resultado<<1;
               if(estado)bit_set(resultado,0);
            }
            output_high(PIN_A4);
            tiempo_espera();
         }
         else{
            if(sensor==4){
               output_high(PIN_C0);
               tiempo_pulso();
               output_low(PIN_C0);
               medida_realizada();
               for(i=0;i<=7;i++)
               {
                  output_high(PIN_C0);
                  tiempo_pulso();
                  output_low(PIN_C0);
                  tiempo_pulso();
                  estado=input(PIN_C1);
                  resultado=resultado<<1;
                  if(estado)bit_set(resultado,0);
               }
               output_high(PIN_C0);
               tiempo_espera();
            }
            else{
               if(sensor==5){
                  output_high(PIN_B1);
                  tiempo_pulso();
                  output_low(PIN_B1);
                  medida_realizada();
                  for(i=0;i<=7;i++)
                  {
                     output_high(PIN_B1);
                     tiempo_pulso();
                     output_low(PIN_B1);
                     tiempo_pulso();
                     estado=input(PIN_B0);
                     resultado=resultado<<1;
                     if(estado)bit_set(resultado,0);
                  }
                  output_high(PIN_B1);
                  tiempo_espera();
               }
               else{
                  if(sensor==6){
                     output_high(PIN_B3);
                     tiempo_pulso();
                     output_low(PIN_B3);
```

```
                        medida_realizada();
                        for(i=0;i<=7;i++)
                        {
                           output_high(PIN_B3);
                           tiempo_pulso();
                           output_low(PIN_B3);
                           tiempo_pulso();
                           estado=input(PIN_B2);
                           resultado=resultado<<1;
                           if(estado)bit_set(resultado,0);

                        }
                        output_high(PIN_B3);
                        tiempo_espera();
                     }
                     else{
                        if(sensor==7){
                           output_high(PIN_B5);
                           tiempo_pulso();
                           output_low(PIN_B5);
                           medida_realizada();
                           for(i=0;i<=7;i++)
                           {
                              output_high(PIN_B5);
                              tiempo_pulso();
                              output_low(PIN_B5);
                              tiempo_pulso();
                              estado=input(PIN_B4);
                              resultado=resultado<<1;
                              if(estado)bit_set(resultado,0);
                           }
                           output_high(PIN_B5);
                           tiempo_espera();
                        }
                        else{
                           if(sensor==8){
                              output_high(PIN_B7);
                              tiempo_pulso();
                              output_low(PIN_B7);
                              medida_realizada();
                              for(i=0;i<=7;i++)
                              {
                                 output_high(PIN_B7);
                                 tiempo_pulso();
                                 output_low(PIN_B7);
                                 tiempo_pulso();
                                 estado=input(PIN_B6);
                                 resultado=resultado<<1;
                                 if(estado)bit_set(resultado,0);
                              }
                              output_high(PIN_B7);
                              tiempo_espera();
                           }
                        }
                     }
                  }
               }
            }
         }
      }
   }
   return (resultado);
}

//Función que genera un ancho de pulso de 0.128ms (anchos del pulso en Vin)
void tiempo_pulso(void)
{
   TMR0 = 254;
```

```
   while(T0IF==0);
   T0IF=0;
}

//Función que genera un tiempo de espera de 1.536ms antes de realizar una
nueva medida*/
void tiempo_espera(void)
{
   TMR0 = 232;
   while (T0IF==0);
   T0IF=0;
}

//Función que genera un tiempo de espera de 70ms en la señal Vin para que el
sensor realice la medida
void medida_realizada(void)
{
   int contador=1;
   T0IF=0;
   TMR0 = 186;
   while(contador<=4)
   {
      while (T0IF==0);
      T0IF=0;
      contador++;
   }
}

//Función que obtiene el valor en cm de la distancia al objeto a partir del
valor proporcionado por el sensor
float convertir_valor(BYTE medida)
{
   float longitud;
   longitud=(15.6/(medida-62))*100;
   return longitud;
}

//Función que calcula el código de comprobación para la transferencia de datos
entre el medidor y la placa de control
void calculo_CRC (void)
{
   BYTE num_bits_1=0;
   int j,i;

   if(operacion==0x03)
   {
        for(i=1;i<=4;i++)
         {
            for(j=0;j<=7;j++)
            {
               if((medida[i]>>j)&1)num_bits_1++;
            }
         }
   }
   else
   {
         for(i=1;i<=8;i++)
         {
            for(j=0;j<=7;j++)
            {
               if((medida[i]>>j)&1)num_bits_1++;
            }
         }
   }
   CRC=num_bits_1;
}
```

TELÉMETRO POR INFRARROJOS PUERTA TRASERA

```
//PROGRAMA TELÉMETRO POR INFRARROJOS PUERTA TRASERA

#include <16F876.h>

#fuses XT, NOLVP, NOWRT, NOBROWNOUT, NOCPD, NOWDT, NOPROTECT
#use delay(clock=4000000)     // Velocidad del oscilador : 4 Mhz
#use RS232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7) //Se definen parámetros para
comunicación RS-232
#use i2c (slave, sda=PIN_C4, scl=PIN_C3, address=0x20, FAST) //Se definen
parámetros para comunicación I2C

#bit R_W = 0x94.2
#bit SSPIF = 0x0C.3
#bit T0IF = 0x0B.2
#byte TMR0 = 0x01
#byte OPTION = 0x81
#byte ADCON1 = 0x9F

BYTE operacion;
BYTE medida[10];
BYTE estado[3];
BYTE CRC;
int flag=0;
int k;
int count=0;

void realizar_operacion(void);
unsigned char GP2D02(int sensor);
void tiempo_pulso(void);
void tiempo_espera(void);
void medida_realizada(void);
void calculo_CRC(void);
void com_i2c(void);
void com_232(void);
float convertir_valor(BYTE medida);

#INT_SSP
void ssp_interrupt ()
{
   if(i2c_poll())
   {
      operacion=i2c_read();
      count++;
   }
   if(R_W==1) //La placa de control quiere recibir datos
   {
      flag=1;
   }
}


void main()
{
   short int comunicacion;
   comunicacion=input(PIN_C2); //Se comprueba el estado del PIN_C2 para saber
si la comunicación se establece con el PC o con la placa de control
   if(comunicacion==0)
   {
      com_i2c(); //La comunicación se establece con la placa de control
   }
   if(comunicacion==1)
   {
      com_232(); //La comunicación se establece con el PC
   }
}

//Función que se ejecuta cuando se establece la comunicación con la placa de
```

```
control
void com_i2c(void)
{
   enable_interrupts(GLOBAL); //Se habilitan todas las interrupciones
   enable_interrupts(INT_SSP);
   while (TRUE)
   {
      while(count!=2);
      count=0;
      realizar_operacion();
      while(flag==0);
      if(operacion==0x05)
      {
         for(k=0;k<=2;k++)
         {
            delay_us(50);
            i2c_write(estado[k]);
         }
      }
      else
      {
         for(k=0;k<=9;k++)
         {
            delay_us(50);
            i2c_write(medida[k]);
         }
      }
      SSPIF=0;
      flag=0;
   }
}

//Función que se ejecuta cuando se establece la comunicación con la placa de
control
void com_232(void)
{
   char seleccion;
   int sensor,auxiliar,i;
   float longitud;

   OPTION = 0b00000101; // Se configura el pre-escaler del timer 0 a 64 y R
pull-up puerto B
   SETUP_ADC_PORTS(NO_ANALOGS);//Puerta A entradas/salidas digitales
   set_tris_a(0x2A); //Se configuran los pines del puerto A como entradas o
salidas
   set_tris_b(0x55); //Se configuran los pines del puerto B como entradas o
salidas
   set_tris_c(0xBE); //Se configuran los pines del puerto C como entradas o
salidas

   output_high(PIN_A0);
   output_high(PIN_A2);
   output_high(PIN_A4);
   output_high(PIN_C0);
   output_high(PIN_B1);
   output_high(PIN_B3);
   output_high(PIN_B5);
   output_high(PIN_B7);

   while(TRUE)
   {
   for(i=0;i<=9;i++)
   {
      medida[i]=0;
   }
   auxiliar=1;
      do
```

```
      {
         printf("\r\nSeleccion de la secuencia de disparo de los sensores de
infrarrojos:\n\n");
         printf("\rPulse la opcion deseada:\n\n");
         printf("\r  1.- Disparos consecutivos de izquierda a derecha\n");
         printf("\r  2.- Disparos consecutivos de derecha a izquierda\n");
         printf("\r  3.- Disparos alternos\n");
         printf("\r  4.- Disparos cada cuatro sensores, uno por
cuadrante\n\n");
         seleccion=getchar();
      }while(seleccion < '1' || seleccion > '4');

   switch(seleccion)
   {
         case '1':
               medida[0]=8;
               for(sensor=1;sensor<=8;sensor++)
               {
                  medida[auxiliar]=GP2D02(sensor);
                  auxiliar++;
               }
               for(auxiliar=1;auxiliar<=8;auxiliar++)
               {
                  if(medida[auxiliar]<=80)
                  {
                    printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",auxiliar+8);
                  }
                  else
                  {
                     longitud=convertir_valor(medida[auxiliar]);
                     printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,auxiliar+8);
                  }
               }
               break;
         case '2':
               medida[0]=8;
               for(sensor=8;sensor>=1;sensor--)
               {
                  medida[auxiliar]=GP2D02(sensor);
                  auxiliar++;
               }
               for(auxiliar=1;auxiliar<=8;auxiliar++)
               {
                  if(medida[auxiliar]<=80)
                  {
                    printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",17-auxiliar);
                  }
                  else
                  {
                     longitud=convertir_valor(medida[auxiliar]);
                     printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,17-auxiliar);
                  }
               }
               break;
         case '3':
               medida[0]=4;
               for(sensor=1;sensor<=8;sensor++)
               {
                  if((sensor==1)||(sensor==3)||(sensor==5)||(sensor==7))
                  {
                     medida[auxiliar]=GP2D02(sensor);
                     auxiliar++;
                  }
```

```
                }
                for(auxiliar=1;auxiliar<=4;auxiliar++)
                {
                   if(medida[auxiliar]<=80)
                   {
                     printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",7+(2*auxiliar));
                   }
                   else
                   {
                      longitud=convertir_valor(medida[auxiliar]);
                      printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,7+(2*auxiliar));
                   }
                }
                break;
          case '4':
                medida[0]=8;
                for(sensor=1;sensor<=8;sensor++)
                {
                   if((sensor==1)||(sensor==5))
                   {
                      medida[auxiliar]=GP2D02(sensor);
                      auxiliar++;
                   }
                }
                auxiliar=3;
                for(sensor=1;sensor<=8;sensor++)
                {
                   if((sensor==2)||(sensor==6))
                   {
                      medida[auxiliar]=GP2D02(sensor);
                      auxiliar++;
                   }
                }
                auxiliar=5;
                for(sensor=1;sensor<=8;sensor++)
                {
                   if((sensor==3)||(sensor==7))
                   {
                      medida[auxiliar]=GP2D02(sensor);
                      auxiliar++;
                   }
                }
                auxiliar=7;
                for(sensor=1;sensor<=8;sensor++)
                {
                   if((sensor==4)||(sensor==8))
                   {
                      medida[auxiliar]=GP2D02(sensor);
                      auxiliar++;
                   }
                }
                for(auxiliar=1;auxiliar<=2;auxiliar++)
                {
                   if(medida[auxiliar]<=80)
                   {
                     printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(4*(auxiliar-1))+9);
                   }
                   else
                   {
                      longitud=convertir_valor(medida[auxiliar]);
                      printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(4*(auxiliar-1))+9);
                   }
                }
```

```
                for(auxiliar=1;auxiliar<=2;auxiliar++)
                {
                   if(medida[auxiliar+2]<=80)
                   {
                     printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(4*(auxiliar-1))+10;

                   }
                   else
                   {
                      longitud=convertir_valor(medida[auxiliar+2]);
                      printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(4*(auxiliar-1))+10);
                   }
                }
                for(auxiliar=1;auxiliar<=2;auxiliar++)
                {
                   if(medida[auxiliar+4]<=80)
                   {
                     printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(4*(auxiliar-1))+11);
                   }
                   else
                   {
                      longitud=convertir_valor(medida[auxiliar+4]);
                      printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(4*(auxiliar-1))+11);
                   }
                }
                for(auxiliar=1;auxiliar<=2;auxiliar++)
                {
                   if(medida[auxiliar+6]<=80)
                   {
                     printf("\rEl objeto se encuentra a +80cm del sensor
%d\n",(4*(auxiliar-1))+12);
                   }
                   else
                   {
                      longitud=convertir_valor(medida[auxiliar+6]);
                      printf("\rEl objeto se encuentra a %3.1fcm del sensor
%d\n",longitud,(4*(auxiliar-1))+12);
                   }
                }
                break;
             default:
                break;
       }
       delay_ms(10);
    }
}

//Función que dispara los sensores y que se ejecuta cuando se establece
comunicación con la placa de control
void realizar_operacion()
{
   short int puerta;
   int sensor, auxiliar=1;
   OPTION = 0b00000101;
   SETUP_ADC_PORTS(NO_ANALOGS);
   DISABLE_INTERRUPTS(INT_EXT); //Se deshabilita la interrupción externa del
pin RB0
   set_tris_a(0x2A);
   set_tris_b(0x55);
   set_tris_c(0xBE);

   output_high(PIN_A0);
   output_high(PIN_A2);
```

```
output_high(PIN_A4);
output_high(PIN_C0);
output_high(PIN_B1);
output_high(PIN_B3);
output_high(PIN_B5);
output_high(PIN_B7);

if(operacion==0x01)
{
      for(sensor=1;sensor<=8;sensor++)
      {
           medida[auxiliar]=GP2D02(sensor);
           auxiliar++;
      }
      medida[0]=8;
      calculo_CRC();
      medida[9]=CRC;
}
if(operacion==0x02)
{
      for(sensor=8;sensor>=1;sensor--)
      {
           medida[auxiliar]=GP2D02(sensor);
           auxiliar++;
      }
      medida[0]=8;
      calculo_CRC();
      medida[9]=CRC;
}
if(operacion==0x03)
{
    for(sensor=1;sensor<=8;sensor++)
    {
        if((sensor==1)||(sensor==3)||(sensor==5)||(sensor==7))
        {
             medida[auxiliar]=GP2D02(sensor);
             auxiliar++;
        }
    }
    medida[0]=4;
    calculo_CRC();
    medida[5]=CRC;
}
if(operacion==0x04)
{
    for(sensor=1;sensor<=8;sensor++)
    {
      if((sensor==1)||(sensor==5)){
          medida[auxiliar]=GP2D02(sensor);
         auxiliar++;
      }
    }
    auxiliar=3;
    for(sensor=1;sensor<=8;sensor++)
    {
      if((sensor==2)||(sensor==6)){
          medida[auxiliar]=GP2D02(sensor);
         auxiliar++;
      }
    }
    auxiliar=5;
    for(sensor=1;sensor<=8;sensor++)
    {
      if((sensor==3)||(sensor==7)){
         medida[auxiliar]=GP2D02(sensor);
         auxiliar++;
      }
```

```
        }
        auxiliar=7;
        for(sensor=1;sensor<=8;sensor++)
        {
          if((sensor==4)||(sensor==8)){
             medida[auxiliar]=GP2D02(sensor);
             auxiliar++;
          }
        }
        medida[0]=8;
        calculo_CRC();
        medida[9]=CRC;
    }
    if(operacion==0x05)
    {
        puerta=input(PIN_C5);
        estado[0]=1;
        estado[1]=puerta;
        estado[2]=estado[1];
    }
}

//Función que genera las señales que posibilitan el disparo de los sensores y
recoge los resultados obtenidos
unsigned char GP2D02(int sensor)
{
   unsigned char resultado;
   int i;
   short int estado;

   resultado=0;

   if(sensor==1){
         output_high(PIN_A0);
         tiempo_pulso();
         output_low(PIN_A0);
         medida_realizada();
         for(i=0;i<=7;i++)
         {
            output_high(PIN_A0);
            tiempo_pulso();
            output_low(PIN_A0);
            tiempo_pulso();
            estado=input(PIN_A1);
            resultado=resultado<<1;
            if(estado)bit_set(resultado,0);
         }
         output_high(PIN_A0);
         tiempo_espera();
   }
   else{
      if(sensor==2){
         output_high(PIN_A2);
         tiempo_pulso();
         output_low(PIN_A2);
         medida_realizada();
         for(i=0;i<=7;i++)
         {
            output_high(PIN_A2);
            tiempo_pulso();
            output_low(PIN_A2);
            tiempo_pulso();
            estado=input(PIN_A3);
            resultado=resultado<<1;
            if(estado)bit_set(resultado,0);
         }
         output_high(PIN_A2);
```

```
      tiempo_espera();
   }
   else{
      if(sensor==3){
         output_high(PIN_A4);
         tiempo_pulso();
         output_low(PIN_A4);
         medida_realizada();
         for(i=0;i<=7;i++)
         {
            output_high(PIN_A4);
            tiempo_pulso();
            output_low(PIN_A4);
            tiempo_pulso();
            estado=input(PIN_A5);
            resultado=resultado<<1;
            if(estado)bit_set(resultado,0);
         }
         output_high(PIN_A4);
         tiempo_espera();
      }
      else{
         if(sensor==4){
            output_high(PIN_C0);
            tiempo_pulso();
            output_low(PIN_C0);
            medida_realizada();
            for(i=0;i<=7;i++)
            {
               output_high(PIN_C0);
               tiempo_pulso();
               output_low(PIN_C0);
               tiempo_pulso();
               estado=input(PIN_C1);
               resultado=resultado<<1;
               if(estado)bit_set(resultado,0);
            }
            output_high(PIN_C0);
            tiempo_espera();
         }
         else{
            if(sensor==5){
               output_high(PIN_B1);
               tiempo_pulso();
               output_low(PIN_B1);
               medida_realizada();
               for(i=0;i<=7;i++)
               {
                  output_high(PIN_B1);
                  tiempo_pulso();
                  output_low(PIN_B1);
                  tiempo_pulso();
                  estado=input(PIN_B0);
                  resultado=resultado<<1;
                  if(estado)bit_set(resultado,0);
               }
               output_high(PIN_B1);
               tiempo_espera();
            }
            else{
               if(sensor==6){
                  output_high(PIN_B3);
                  tiempo_pulso();
                  output_low(PIN_B3);
                  medida_realizada();
                  for(i=0;i<=7;i++)
                  {
```

```
                        output_high(PIN_B3);
                        tiempo_pulso();
                        output_low(PIN_B3);
                        tiempo_pulso();
                        estado=input(PIN_B2);
                        resultado=resultado<<1;
                        if(estado)bit_set(resultado,0);

                     }
                     output_high(PIN_B3);
                     tiempo_espera();
                  }
                  else{
                     if(sensor==7){
                        output_high(PIN_B5);
                        tiempo_pulso();
                        output_low(PIN_B5);
                        medida_realizada();
                        for(i=0;i<=7;i++)
                        {
                           output_high(PIN_B5);
                           tiempo_pulso();
                           output_low(PIN_B5);
                           tiempo_pulso();
                           estado=input(PIN_B4);
                           resultado=resultado<<1;
                           if(estado)bit_set(resultado,0);
                        }
                        output_high(PIN_B5);
                        tiempo_espera();
                     }
                     else{
                        if(sensor==8){
                           output_high(PIN_B7);
                           tiempo_pulso();
                           output_low(PIN_B7);
                           medida_realizada();
                           for(i=0;i<=7;i++)
                           {
                              output_high(PIN_B7);
                              tiempo_pulso();
                              output_low(PIN_B7);
                              tiempo_pulso();
                              estado=input(PIN_B6);
                              resultado=resultado<<1;
                              if(estado)bit_set(resultado,0);
                           }
                           output_high(PIN_B7);
                           tiempo_espera();
                        }
                     }
                  }
               }
            }
         }
      }
   }
   return (resultado);
}

//Función que genera un ancho de pulso de 0.128ms (anchos del pulso en Vin)
void tiempo_pulso(void)
{
   TMR0 = 254;
   while(T0IF==0);
   T0IF=0;
}
```

```
//Función que genera un tiempo de espera de 1.536ms antes de realizar una
nueva medida*/
void tiempo_espera(void)
{
   TMR0 = 232;
   while (T0IF==0);
   T0IF=0;
}

//Función que genera un tiempo de espera de 70ms en la señal Vin para que el
sensor realice la medida
void medida_realizada(void)
{
   int contador=1;
   T0IF=0;
   TMR0 = 186;
   while(contador<=4)
   {
        while (T0IF==0);
      T0IF=0;
      contador++;
   }
}

//Función que obtiene el valor en cm de la distancia al objeto a partir del
valor proporcionado por el sensor
float convertir_valor(BYTE medida)
{
   float longitud;
   longitud=(15.6/(medida-62))*100;
   return longitud;
}

//Función que calcula el código de comprobación para la transferencia de datos
entre el medidor y la placa de control
void calculo_CRC (void)
{
   BYTE num_bits_1=0;
   int j,i;

   if(operacion==0x03)
   {
        for(i=1;i<=4;i++)
         {
            for(j=0;j<=7;j++)
            {
               if((medida[i]>>j)&1)num_bits_1++;
            }
         }
   }
   else
   {
         for(i=1;i<=8;i++)
         {
            for(j=0;j<=7;j++)
            {
               if((medida[i]>>j)&1)num_bits_1++;
            }
         }
   }
   CRC=num_bits_1;
}
```

PLACA DE CONTROL

```
//PROGRAMA QUE DISPARA LOS SENSORES DE INFRARROJOS,LOS SONAR DE LA PUERTA
DELANTERA Y TRASERA Y LA BRÚJULA MEDIANTE COMUNICACIÓN I2C
//MUESTRA LOS RESULTADOS POR EL PC

#include <18F2550.h>


#fuses
XT,NOLVP,STVREN,NOWRTC,NOWDT,NOPROTECT,NOCPB,NOWRT,NOEBTR,NOWRTB,NOEBTRB,NOBRO
WNOUT,NOPUT,DEBUG,NOWRTD,NOWRTB,NOCPD,NOXINST
#use delay(clock=4000000)
#use RS232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use I2C (master, sda=PIN_B0, scl=PIN_B1, FAST)

void write_slave(byte i2c_direccion,byte i2c_datos);
void read_slave(byte i2c_direccion);
void retardo(int n);
void read_slave_assign(byte i2c_direccion);
void enable_puerta_delantera(void);
void enable_puerta_trasera(void);
void comprobar_asignacion_IR_delantero_trasero(void);
void comprobar_asignacion_sonar_delantero_trasero(void);
void asignacion_puerta_delantera(void);
void asignacion_puerta_trasera(void);
void disparar_puerta_A(void);
void disparar_puerta_B(void);
void disparar_puerta_A_y_B(void);
void disparar_IR_delantero(void);
void disparar_IR_trasero(void);
void disparar_sonar_delantero(void);
void disparar_sonar_trasero(void);
void CMPS03_Read(void);
void enviar_trama(void);
void calculo_CRC(void);

#define CMPS03 0xC0

#byte PORTB = 0xF81

BYTE medida[10],medida_IR[10],medida_sonar[10];//variable donde se guarda el
resultado proporcionado por el sensor de IR o el sonar
BYTE asignacion[3];
BYTE orientacion[2];
BYTE error[17];
BYTE i2c_direccion_IR_delantero=0x10,i2c_direccion_IR_trasero=0x20;
BYTE i2c_direccion_sonar_delantero=0x30,i2c_direccion_sonar_trasero=0x40;
BYTE i2c_datos,CRC;
BYTE
puerta_delantera,puerta_trasera,puerta_IR_delantera,puerta_IR_trasera,puerta_s
onar_delantera,puerta_sonar_trasera;
int i,j,k=0,n;
unsigned char selec_sensor,puerta,secuencia_disparo,ready;
int1 flag_puerta_delantera,flag_puerta_trasera,flag_error;
int1 enable_IR_delantero=1,enable_IR_trasero=1; //habilita el disparo de los
sensores de la puerta 1 y la puerta 2
int1 enable_sonar_delantero=1,enable_sonar_trasero=1,enable_brujula=1;
//habilita el disparo de los sensores de la puerta 1 y la puerta 2
boolean ACK;


void main()
{
   DISABLE_INTERRUPTS(GLOBAL);
   port_b_pullups(TRUE);//para los leds de la placa
   set_tris_b(0x07);
   set_tris_c(0xBF);
```

```
////////////////////////////////////////////////////////////////////////
//////////////INICIALIZACIÓN////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
   output_high(PIN_B1);
   output_high(PIN_B0);
   delay_ms(10);


   output_low(PIN_B7);
   output_low(PIN_B6);
   output_low(PIN_B5);
   output_low(PIN_B4);
   output_low(PIN_B3);
   flag_error=0;

   for(i=0;i<=9;i++)
   {
      medida[i]=0;
   }
   for(i=0;i<=9;i++)
   {
      medida_IR[i]=0;
   }
   for(i=0;i<=9;i++)
   {
      medida_sonar[i]=0;
   }
   for(i=0;i<=1;i++)
   {
      orientacion[i]=0;
   }
   for(i=0;i<=2;i++)
   {
       asignacion[i]=0;
   }
   for(i=0;i<=16;i++)
   {
       error[i]=0;
   }
   ready=getchar();
   if(ready=='r')
   {
         //COMPROBACION ASIGNACION
         asignacion_puerta_delantera(); /*comprobar si la puerta delantera es
la puerta A o B*/
         if(flag_error==1)
         {
            enviar_trama();
            restart_cause();
         }

         asignacion_puerta_trasera(); /*comprobar si la puerta trasera es la
puerta A o B*/
         if(flag_error==1)
         {
            enviar_trama();
            restart_cause();
         }

      /*COMPROBAR SI ASIGNACION IR DELANTERO Y TRASERO ES LA MISMA*/
         comprobar_asignacion_IR_delantero_trasero();
         if(flag_error==1)
         {
            enviar_trama();
            restart_cause();
         }
```

```
      /*COMPROBAR SI ASIGNACION SONAR DELANTERO Y TRASERO ES LA MISMA*/
         comprobar_asignacion_sonar_delantero_trasero();
         if(flag_error==1)
         {
            enviar_trama();
            restart_cause();
         }

      //enviar datos de la inicialización para determinar dispositivos
disponibles
         enviar_trama();
   }
   else
   {
       error[16]=1;
       enviar_trama();
       restart_cause();
   }



////////////////////////////////////////////////////////////////////////////
//////////////////////TOMA DE MEDIDAS///////////////////////////////////////
////////////////////////////////////////////////////////////////////////////


   while(TRUE)
   {

   flag_puerta_delantera=0;
   flag_puerta_trasera=0;
   flag_error=0;
   ready='x';

   for(i=0;i<=16;i++)
   {
      error[i]=0;
   }

   ready=getchar();
   if(ready=='r')
   {
      enable_puerta_delantera();
      if(flag_error==1)
      {
         enviar_trama();
         restart_cause();
      }
      enable_puerta_trasera();
      if(flag_error==1)
      {
         enviar_trama();
         restart_cause();
      }
      enviar_trama();
      selec_sensor=getchar();
      puerta=getchar();
      secuencia_disparo=getchar();

      if(secuencia_disparo=='1')i2c_datos=0x01;
      if(secuencia_disparo=='2')i2c_datos=0x02;
      if(secuencia_disparo=='3')i2c_datos=0x03;
      if(secuencia_disparo=='4')i2c_datos=0x04;

      if(enable_brujula==1)
      {
```

```
            CMPS03_Read();
            if(error[13]==1)
            {
               enviar_trama();
               continue;
            }
         }

         if((puerta =='A')||(puerta=='a'))
         {
            disparar_puerta_A();
            if((error[5]==1)||(error[6]==1)||(error[7]==1)||(error[8]==1))
            {
               enviar_trama();
               continue;
            }
            enviar_trama();
         }
         if((puerta=='B')||(puerta=='b'))
         {
            disparar_puerta_B();
            if((error[5]==1)||(error[6]==1)||(error[7]==1)||(error[8]==1))
            {
               enviar_trama();
               continue;
            }
            enviar_trama();
         }
      }
      else
      {
          error[16]=1;
          enviar_trama();
          restart_cause();
      }
      }
}


/*----FUNCIÓN QUE COMPRUEBA QUE AL MENOS UNO DE LOS SENSORES (IR O SONAR) DE
LA PUERTA DELANTERA ESTÉN HABILITADOS----*/
void enable_puerta_delantera(void)
{
   if((enable_IR_delantero==0)&&(enable_sonar_delantero==0))
   {
      error[0]=1;
      flag_error=1;
   }
}


/*----FUNCIÓN QUE COMPRUEBA QUE AL MENOS UNO DE LOS SENSORES (IR O SONAR) DE
LA PUERTA DELANTERA ESTÉN HABILITADOS----*/
void enable_puerta_trasera(void)
{
   if((enable_IR_trasero==0)&&(enable_sonar_trasero==0))
   {
      error[1]=1;
      flag_error=1;
   }
}


/*----FUNCIÓN PARA SABER CUAL ES LA ASIGNACION DE LOS SENSORES DE LA PUERTA
DELANTERA----*/
void asignacion_puerta_delantera(void)
{
```

```
//comunicarse con IR delantero.......operacion=5

     i2c_datos=0x05;
     n=4; //retardo de 200ms
     output_high(PIN_B5);
     output_high(PIN_B3);
     write_slave(i2c_direccion_IR_delantero,i2c_datos);
     if(ACK)
     {
          error[5]=1;
          enable_IR_delantero=0;
     }
     if(enable_IR_delantero==1)
     {
          read_slave_assign(i2c_direccion_IR_delantero);
          if(ACK)
          {
               error[5]=1;
               enable_IR_delantero=0;
          }
          if(enable_IR_delantero==1)
          {
              if(asignacion[j]!=asignacion[1])
              {
                  error[5]=1;
                  error[4]=1;
                  enable_IR_delantero=0;
              }
          }
       }
       if(enable_IR_delantero==1)
       {
            puerta_IR_delantera=asignacion[1];
       }
       output_low(PIN_B5);
       output_low(PIN_B3);


//comunicarse con sonar delantero........operacion=5

       output_high(PIN_B6);
       output_high(PIN_B3);
       write_slave(i2c_direccion_sonar_delantero,i2c_datos);
       if(ACK)
       {
            error[6]=1;
            enable_sonar_delantero=0;
       }
       if(enable_sonar_delantero==1)
       {
            read_slave_assign(i2c_direccion_sonar_delantero);
            if(ACK)
            {
               error[6]=1;
               enable_sonar_delantero=0;
            }
            if(enable_sonar_delantero==1)
            {
               if(asignacion[j]!=asignacion[1])
               {
                  error[6]=1;
                  error[4]=1;
                  enable_sonar_delantero=0;
               }
            }
       }
       if(enable_sonar_delantero==1)
```

```
        {
             puerta_sonar_delantera=asignacion[1];
        }
        output_low(PIN_B6);
        output_low(PIN_B3);

//comprobar misma asignacion IR_delantero y sonar_delantero

      if((enable_IR_delantero==1)&&(enable_sonar_delantero==1))
      {
         if(puerta_IR_delantera!=puerta_sonar_delantera)
         {
            error[9]=1;
            flag_error=1;
            return;
         }
         puerta_delantera=asignacion[1];
      }
      else
      {
         if(enable_IR_delantero==1)
         {
            puerta_delantera=puerta_IR_delantera;
         }
         if(enable_sonar_delantero==1)
         {
            puerta_delantera=puerta_sonar_delantera;
         }
      }
}

/*----FUNCIÓN PARA DISPARAR LOS SENSORES IR PUERTA DELANTERA------*/
void disparar_IR_delantero(void)
{
   output_high(PIN_B5);
   output_high(PIN_B3);
   write_slave(i2c_direccion_IR_delantero,i2c_datos);
   if(ACK)
   {
        error[5]=1;
        enable_IR_delantero=0;
        return;
   }
   read_slave(i2c_direccion_IR_delantero);
   if(ACK)
   {
        error[5]=1;
        enable_IR_delantero=0;
        return;
   }
   calculo_CRC();
   if(medida[k]!=CRC)
   {
        error[5]=1;
        error[4]=1;
        enable_IR_delantero=0;
        return;
   }
   output_low(PIN_B5);
   output_low(PIN_B3);
}

/*----FUNCIÓN PARA INDICAR AL DISPOSITIVO ESCLAVO LA OPERACION A REALIZAR----
*/
void write_slave(byte i2c_direccion, byte i2c_datos)
{
   i2c_start();
```

```
   ACK=i2c_write(i2c_direccion+0); //envio la direccion del esclavo (sensores
IR) + el bit R/W = 0 (escribir)
                           // me aseguro de recibir el ACK. También se puede
escribir como (DIREC+0)
   if (!ACK)
   {
      i2c_write(i2c_datos);   //envío al esclavo una orden para que ejecute
una operacion
   }
   if(ACK)//OK
   {
      error[2]=1;
   }
   i2c_stop();
   retardo(n);
}


/*----FUNCIÓN PARA LEER LOS DATOS ENVIADOS POR EL DISPOSITIVO ESCLAVO------*/
void read_slave(byte i2c_direccion)
{
   i2c_start();   //Repetición de start
   ACK=i2c_write(i2c_direccion+1);      //Dirección del sensor + bit de R/W =
1 (leer)
   if(!ACK)
   {
      medida[0]=i2c_read();   //Recogemos lectura y colocamos NO ACK
      for(k=1;k<=medida[0];k++)
      {
         medida[k]=i2c_read();   //Recogemos lectura y colocamos NO ACK
      }
      medida[k]=i2c_read(0);
   }
   if(ACK)
   {
        error[3]=1;
   }
   i2c_stop();
}


/*----FUNCIÓN PARA LEER LA ASIGNACIÓN DEL DISPOSITIVO ESCLAVO----*/
void read_slave_assign(byte i2c_direccion)
{
   i2c_start();   //Repetición de start
   ACK=i2c_write(i2c_direccion+1);      //Dirección del sensor + bit de R/W =
1 (leer)
   if(!ACK)
   {
      asignacion[0]=i2c_read();   //Recogemos lectura y colocamos NO ACK
      for(j=1;j<=asignacion[0];j++)
      {
         asignacion[j]=i2c_read();   //Recogemos lectura y colocamos NO ACK
      }
      asignacion[j]=i2c_read(0);
   }
   if(ACK)
   {
        error[3]=1;
   }
   i2c_stop();
}


/*----FUNCIÓN PARA SABER CUAL ES LA ASIGNACION DE LOS SENSORES DE LA PUERTA
TRASERA----*/
void asignacion_puerta_trasera(void)
```

```
{

//comunicarse con IR trasero.......operacion=5
      i2c_datos=0x05;
      n=4; //retardo de 200ms
      output_high(PIN_B5);
      output_high(PIN_B4);
      write_slave(i2c_direccion_IR_trasero,i2c_datos);
      if(ACK)
      {
          error[7]=1;
          enable_IR_trasero=0;
      }
      if(enable_IR_trasero==1)
      {
         read_slave_assign(i2c_direccion_IR_trasero);
         if(ACK)
         {
             error[7]=1;
             enable_IR_trasero=0;
         }
         if(enable_IR_trasero==1)
         {
             if(asignacion[j]!=asignacion[1])
             {
                 error[7]=1;
                 error[4]=1;
                 enable_IR_trasero=0;
             }
         }
      }
      if(enable_IR_trasero==1)
      {
         puerta_IR_trasera=asignacion[1];
      }
      output_low(PIN_B5);
      output_low(PIN_B4);


 //comunicarse con sonar trasero........operacion=5
      output_high(PIN_B6);
      output_high(PIN_B4);
      write_slave(i2c_direccion_sonar_trasero,i2c_datos);
      if(ACK)
      {
          error[8]=1;
          enable_sonar_trasero=0;
      }
      if(enable_sonar_trasero==1)
      {
         read_slave_assign(i2c_direccion_sonar_trasero);
         if(ACK)
         {
            error[8]=1;
            enable_sonar_trasero=0;
         }
         if(enable_sonar_trasero==1)
         {
            if(asignacion[j]!=asignacion[1])
            {
               error[8]=1;
               error[4]=1;
               enable_sonar_trasero=0;
            }
         }
      }
      if(enable_sonar_trasero==1)
```

```
      {
         puerta_sonar_trasera=asignacion[1];
      }
      output_low(PIN_B6);
      output_low(PIN_B4);


//comprobar misma asignacion IR_trasero y sonar_trasero
   if((enable_IR_trasero==1)&&(enable_sonar_trasero==1))
   {
      if(puerta_IR_trasera!=puerta_sonar_trasera)
      {
          error[10]=1;
          flag_error=1;
          return;
      }
      puerta_trasera=asignacion[1];
   }
   else
   {
      if(enable_IR_trasero==1)
      {
         puerta_trasera=puerta_IR_trasera;
      }
      if(enable_sonar_trasero==1)
      {
         puerta_trasera=puerta_sonar_trasera;
      }
   }
}


/*----FUNCIÓN PARA DISPARAR LOS SENSORES IR PUERTA TRASERA------*/
void disparar_IR_trasero(void)
{
   output_high(PIN_B5);
   output_high(PIN_B4);
   write_slave(i2c_direccion_IR_trasero,i2c_datos);
   if(ACK)
   {
         error[7]=1;
         enable_IR_trasero=0;
         return;
   }
   read_slave(i2c_direccion_IR_trasero);
   if(ACK)
   {
         error[7]=1;
         enable_IR_trasero=0;
         return;
   }
   calculo_CRC();
   if(medida[k]!=CRC)
   {
         error[7]=1;
         error[4]=1;
         enable_IR_trasero=0;
         return;
   }
   output_low(PIN_B5);
   output_low(PIN_B4);
}


/*----FUNCIÓN QUE COMPRUEBA QUE LA ASIGNACION DE LOS IR DE LAS PUERTAS ES
DISTINTA----*/
void comprobar_asignacion_IR_delantero_trasero(void)
```

```
{
   if((enable_IR_delantero==1)&&(enable_IR_trasero==1))
   {
      if(puerta_IR_delantera==puerta_IR_trasera)
      {
         error[11]=1;
         flag_error=1;
      }
   }
}


/*----FUNCIÓN QUE COMPRUEBA QUE LA ASIGNACION DE LOS SONAR DE LAS PUERTAS ES
DISTINTA----*/
void comprobar_asignacion_sonar_delantero_trasero(void)
{
   if((enable_sonar_delantero==1)&&(enable_sonar_trasero==1))
   {
      if(puerta_sonar_delantera==puerta_sonar_trasera)
      {
         error[12]=1;
         flag_error=1;
      }
   }
}


/*----FUNCIÓN PARA DISPARAR LA BRÚJULA Y LEER LA ORIENTACIÓN------*/
void CMPS03_Read(void)
{
   output_high(PIN_B7);
   i2c_start();
   ACK=i2c_write(CMPS03);
   if(!ACK)
   {
      i2c_write(0x2);
      i2c_stop();
      i2c_start();
      i2c_write(CMPS03+1);
      orientacion[0]=i2c_read();
      orientacion[1]=i2c_read(0);
   }
   if(ACK)
   {
      error[2]=1;
      error[13]=1;
      enable_brujula=0;
   }
   i2c_stop();
   output_low(PIN_B7);
}


/*----FUNCIÓN PARA DISPARAR LA PUERTA A----------------*/
void disparar_puerta_A (void)
{
   if(puerta_delantera==0) //La puerta delantera (sensores del 1 al 8) es la
puerta A
   {
      if((selec_sensor=='2')||(selec_sensor=='3'))
      {
         if(enable_IR_delantero==1)
         {
            n=13; //retardo de 650ms
            disparar_IR_delantero();
            if(error[5]==1)return;
            for(i=0;i<=9;i++)
```

```
            {
               medida_IR[i]=medida[i];
            }
            flag_puerta_delantera=1; //para saber que se han disparado los
sensores de la puerta delantera
         }
      }
      if((selec_sensor=='1')||(selec_sensor=='3'))
      {
         if(enable_sonar_delantero==1)
         {
            n=7; //retardo de 350ms
            disparar_sonar_delantero();
            if(error[6]==1)return;
            for(i=0;i<=9;i++)
            {
               medida_sonar[i]=medida[i];
            }
            flag_puerta_delantera=1; //para saber que se han disparado los
sensores de la puerta delantera
         }
      }
   }
   if(puerta_trasera==0) //la puerta trasera (sensores del 9 al 16) es la
puerta A
   {
      if((selec_sensor=='2')||(selec_sensor=='3'))
      {
         if(enable_IR_trasero==1)
         {
            n=13; //retardo de 650ms
            disparar_IR_trasero();
            if(error[7]==1)return;
            for(i=0;i<=9;i++)
            {
               medida_IR[i]=medida[i];
            }
            flag_puerta_trasera=1;
         }
      }
      if((selec_sensor=='1')||(selec_sensor=='3'))
      {
         if(enable_sonar_trasero==1)
         {
            n=7; //retardo de 350ms
            disparar_sonar_trasero();
            if(error[8]==1)return;
            for(i=0;i<=9;i++)
            {
               medida_sonar[i]=medida[i];
            }
            flag_puerta_trasera=1;
         }
      }
   }
}

/*----FUNCIÓN RETARDO PARA ESPERAR LA LECTURA DE LOS DATOS DE LOS DISPOSITIVOS
ESCLAVOS-----*/
void retardo(int n)
{
   for(;n!=0;n--)
   {
      delay_ms(50);
   }
}
```

```
/*----FUNCIÓN PARA ENVIAR TRAMA AL PC POR EL PUERTO USB-------*/
void enviar_trama(void)
{
   for(i=0;i<=16;i++)
   {
      putchar(error[i]);
   }
   for(i=0;i<=1;i++)
   {
      putchar(orientacion[i]);
   }
   for(i=0;i<=9;i++)
   {
      putchar(medida_IR[i]);
   }
   for(i=0;i<=9;i++)
   {
         putchar(medida_sonar[i]);
   }
   putchar(flag_puerta_delantera);
   putchar(flag_puerta_trasera);
   putchar(k); //identifica el último byte recibido desde los telémetros
}


/*----FUNCIÓN PARA DISPARAR LOS SENSORES SONAR PUERTA DELANTERA------*/
void disparar_sonar_delantero(void)
{
   output_high(PIN_B6);
   output_high(PIN_B3);
   write_slave(i2c_direccion_sonar_delantero,i2c_datos);
   if(ACK)
   {
       error[6]=1;
       enable_sonar_delantero=0;
       return;
   }
   read_slave(i2c_direccion_sonar_delantero);
   if(ACK)
   {
       error[6]=1;
       enable_sonar_delantero=0;
       return;
   }
   calculo_CRC();
   if(medida[k]!=CRC)
   {
       error[6]=1;
       error[4]=1;
       enable_sonar_delantero=0;
       return;
   }
   output_low(PIN_B6);
   output_low(PIN_B3);
}


/*----FUNCIÓN PARA DISPARAR LOS SENSORES SONAR PUERTA TRASERA------*/
void disparar_sonar_trasero(void)
{
   output_high(PIN_B6);
   output_high(PIN_B4);
   write_slave(i2c_direccion_sonar_trasero,i2c_datos);
   if(ACK)
   {
       error[8]=1;
       enable_sonar_trasero=0;
```

```
        return;
    }
    read_slave(i2c_direccion_sonar_trasero);
    if(ACK)
    {
        error[8]=1;
        enable_sonar_trasero=0;
        return;
    }
    calculo_CRC();
    if(medida[k]!=CRC)
    {
        error[8]=1;
        error[4]=1;
        enable_sonar_trasero=0;
        return;
    }
    output_low(PIN_B6);
    output_low(PIN_B4);
}



/*-----FUNCIÓN PARA DISPARAR LA PUERTA B----------------*/
void disparar_puerta_B(void)
{
    if(puerta_delantera==1) //La puerta delantera es la puerta B
    {
       if((selec_sensor=='2')||(selec_sensor=='3'))
       {
          if(enable_IR_delantero==1)
          {
             n=13; //retardo de 650ms
             disparar_IR_delantero();
             if(error[5]==1)return;
             for(i=0;i<=9;i++)
             {
                medida_IR[i]=medida[i];
             }
             flag_puerta_delantera=1; //para saber que se han disparado los
sensores de la puerta delantera
          }
       }
       if((selec_sensor=='1')||(selec_sensor=='3'))
       {
          if(enable_sonar_delantero==1)
          {
             n=7; //retardo de 350ms
             disparar_sonar_delantero();
             if(error[6]==1)return;
             for(i=0;i<=9;i++)
             {
                medida_sonar[i]=medida[i];
             }
             flag_puerta_delantera=1; //para saber que se han disparado los
sensores de la puerta delantera
          }
       }
    }
    if(puerta_trasera==1) //la puerta trasera es la puerta B
    {
       if((selec_sensor=='2')||(selec_sensor=='3'))
       {
          if(enable_IR_trasero==1)
          {
             n=13; //retardo de 650ms
             disparar_IR_trasero();
```

```
            if(error[7]==1)return;
            for(i=0;i<=9;i++)
            {
               medida_IR[i]=medida[i];
            }
            flag_puerta_trasera=1;
         }
      }
      if((selec_sensor=='1')||(selec_sensor=='3'))
      {
         if(enable_sonar_trasero==1)
         {
            n=7; //retardo de 350ms
            disparar_sonar_trasero();
            if(error[8]==1)return;
            for(i=0;i<=9;i++)
            {
               medida_sonar[i]=medida[i];
            }
            flag_puerta_trasera=1;
         }
      }
   }
}


//FUNCIÓN PARA CALCULAR EL CÓDICO DE COMPROBACIÓN
void calculo_CRC (void)
{
   BYTE num_bits_1=0;
   int j,i;

   if(i2c_datos==0x03)
   {
        for(i=1;i<=4;i++)
        {
            for(j=0;j<=7;j++)
            {
               if((medida[i]>>j)&1)num_bits_1++;
            }
        }
   }
   else
   {
         for(i=1;i<=8;i++)
         {
            for(j=0;j<=7;j++)
            {
               if((medida[i]>>j)&1)num_bits_1++;
            }
         }
   }
   CRC=num_bits_1;
}
```

ANEXO D: FUNCIONES Y DIRECTIVAS DEL COMPILADOR CCS

Función i2c_start ( ): Genera la condición de inicio en el modo maestro de I^2C. Se tiene que utilizar junto a la directiva #use i2c.

Función i2c_stop ( ): Genera la condición de parada en el modo maestro de I^2C. Se tiene que utilizar junto a la directiva #use i2c.

Función i2c_write (data): Data es un entero de 8 bits. Envía un byte al bus I^2C y devuelve un bit de valor lógico '0' (ACK) o '1' (NO ACK) de confirmación. No tiene fijado tiempo máximo de espera. Se tiene que utilizar junto a la directiva #use i2c.

Función i2c_read ( ): Lee un byte del bus I^2C. No tiene fijado tiempo máximo de espera. Se tiene que utilizar junto a la directiva #use i2c. Se puede utilizar el parámetro optativo '0' para que la función no envíe el carácter acuse de recibo (ACK), de los datos recibidos.

Función i2c_poll ( ): Esta función retorna un valor distinto de cero cuando el hardware ha recibido un byte en el buffer. En ese momento se produce una llamada a la función i2c_read que devolverá inmediatamente el byte recibido.

Función enable_interrupts (level): Esta función activa la interrupción del nivel dado en level. Queda a cargo del programador definir un procedimiento o rutina de atención, para el caso que se produzca la interrupción indicada. El nivel GLOBAL permite todas las interrupciones que estén habilitadas de forma individual. Los niveles de interrupción, por ejemplo, pueden ser:

- GLOBAL (habilitar todas las interrupciones)
- INT_EXT (habilitar la interrupción externa)
- INT_TIMER1 (habilitar la interrupción del timer1).
- INT_I2C (habilitar la interrupción I^2C).

Función disable_interrupts (level): Esta función desactiva la interrupción del nivel dado en level. El nivel GLOBAL prohíbe todas las interrupciones, aunque estén habilitadas o permitidas. Los niveles de interrupción pueden ser los mismos que para la función enable_interrupts.

Función output_low (pin): Esta función pone a nivel lógico '0' el pin determinado en el parámetro.

Función output_high (pin): Esta función pone a nivel lógico '1' el pin determinado en el parámetro.

Función input (pin): Devuelve un entero corto, indicando el estado del pin indicado en el parámetro.

Función port_b_pullups (flag): Esta función activa/desactiva las resistencias pullups en las entradas del puerto B. Flag puede ser TRUE (activa) o FALSE (desactiva).

Función set_tris_a (value): Estas funciones permiten escribir directamente los registros tri-estado para la configuración del puerto A. Cada bit de value representa una patilla. Un '1' indica que la patilla es de entrada y un '0' que es de salida. También se puede realizar esta operación en el resto de puertos.

Función delay_ms (time): Esta función realiza retardos del valor especificado en time. Dicho valor de tiempo es en milisegundos y el rango es 0-65535. Es preciso utilizar la directiva #use delay(clock) antes de la llamada a esta función, para que el compilador sepa la frecuencia de reloj.

Función restart_cause ( ): Esta función devolverá la razón por la que se ha producido el último reset del procesador. Los valores de retorno pueden ser: WDT_FROM_SLEEP, WDT_TIMEOUT, MCLR_FROM_SLEEP, NORMAL_POWER_UP.

Directiva BIT, sintaxis: #bit id = x.y: Se crea una nueva variable con nombre id, en la posición de memoria x y bit y.

Directiva BYTE, sintaxis: #byte id = x: Asigna a la variable con nombre id una posición de memoria x.

Directiva DEFINE, sintaxis #define id text: Asigna el valor de text al identificador id.

Directiva FUSES, sintaxis #fuse opciones: Define los fuses que serán empleados en la programación. No afecta a la compilación ni a los archivos de salida. Algunas opciones pueden ser: XT, NOWDT, NOPROTECT, NOPUT, NOBROWNOUT…

Directiva INCLUDE, sintaxis #include <filename>: Filename es el nombre de un archivo válido. Puede incluir código normal o direcciones o rutas de información.

Directiva USE DELAY, sintaxis #use delay (clock=frecuencia): Determina la frecuencia de la señal de reloj. El valor de la frecuencia puede ser de 1 a 100.000.000 (1Hz-100MHz).

Directiva USE I2C, sintaxis #use i2c (opciones): Determina los parámetros de la comunicación I^2C, asignando el maestro y el esclavo, pines del módulo MSSP, dirección del esclavo, velocidad de transmisión de los datos…

Directiva USE RS232, sintaxis #use rs232 (opciones): Determina los parámetros de la comunicación RS232, pines del módulo USART, velocidad de transmisión de los datos…

ANEXO E: GUÍA DE FUNCIONAMIENTO

Antes de iniciar el funcionamiento del sistema, hay que colocar correctamente los jumpers de las placas que definen la denominación de puerta (A o B) y el tipo de comunicación (serie o I^2C). Se tiene que recordar que las placas de un mismo tipo de sensor tienen que tener denominaciones de puerta diferentes y que las placas de una misma puerta tienen que tener la misma denominación.

Una vez realizada la configuración se pone en funcionamiento el sistema. El PC del robot envía un carácter de reconocimiento a la placa de control, para determinar que la placa se encuentra disponible y que la comunicación entre ambos dispositivos es correcta. En caso contrario la placa envía el error correspondiente al PC y el sistema se para.

Si el carácter es recibido de forma correcta, se inicia el proceso de reconocimiento de dispositivos por parte de la placa de control. Este reconocimiento se realiza una vez durante todo el funcionamiento del sistema. En este proceso se determina qué dispositivos se encuentran conectados al bus I^2C y la asignación de puerta de cada uno, así como las diferentes configuraciones. Si se realizó el primer paso de esta guía correctamente, no se producirán errores que hagan parar el sistema, si no es así, se generará un error y el sistema se detendrá.

Una vez finalizado el paso anterior se notifican los dispositivos que no estén conectados y se inicia la toma de medidas. En esta fase se selecciona el sensor, la puerta y la secuencia de disparo de los sensores y se transmiten estos datos a la placa de control. Ésta se encarga de disparar el telémetro seleccionado así como la brújula para determinar la posición del robot, los resultados obtenidos son enviados al PC del robot. Tanto en la placa de control como en el PC del robot, cuando se reciben datos, se activa un código de comprobación para determinar que los datos se han recibido de forma correcta, en caso de fallo no se recogen los datos. Si alguno de los telémetros o brújula no estuviera disponible por cualquier razón, se deshabilita el dispositivo y se notifica el error, vigilando que no queden fuera de servicio los telémetros de una misma puerta (en este caso el sistema se para). Una vez recibidos los datos desde la placa de control, se calculan las distancias en cm y la orientación del robot y se muestran los resultados por pantalla. Seguidamente se puede repetir el proceso de toma de medidas las veces que se consideren necesarias.

BIBLIOGRAFÍA

[1] http://roboticslab.uc3m.es/maggie/maggietutorial Visitada por última vez en Mayo 2009.

[2] http://www.ifr.org Visitada por última vez en Mayo 2009.

[3] http://www.worldrobotics.org Visitada por última vez en Mayo 2009.

[4] http://es.wikipedia.org/wiki/Sensor#Tipos_de_sensores Visitada por última vez en Mayo 2009.

[5] http://www.x-robotics.com/sensores.htm Visitada por última vez en Mayo 2009

[6] http://www.infoab.uclm.es/labelec/Solar/Otros/Infrarrojos/sensor_sharp_gp2d02.htm Visitada por última vez en Mayo 2009

[7] http://www.superrobotica.com/S320160.htm Visitada por última vez en Mayo 2009.

[8] http://www.datasheetcatalog.net/es/datasheets_pdf/C/M/P/S/CMPS03.shtml Visitada por última vez en Mayo 2009.

[9] http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf Visitada por última vez en Mayo 2009.

[10] http://ww1.microchip.com/downloads/en/DeviceDoc/39632D.pdf Visitada por última vez en Mayo 2009.

[11] http://www.mcc-us.com/i2cHowToUseIt1995.pdf Visitada por última vez en Mayo 2009.

[12] http://www.barello.net/Papers/GP2D02/ Visitada por última vez en Mayo 2009.

[13] http://es.wikipedia.org/wiki/Reflectividad Visitada por última vez en Mayo 2009.

[14] Amorós O. (2009). *Sistema sensorial para robots móviles basado en multiplexación de sensores sonar*. Proyecto fin de carrera. Universidad Carlos III de Madrid, Escuela Politécnica Superior.

- Angulo, J.M. (2007). *Microcontroladores PIC*. Madrid: McGraw-Hill.

- Gottfried, B. (1999). Programación en C. Madrid: McGraw-Hill.

- Fernández, C. Pérez, I. y Vergaz, B (2006). *Manual Pspice y layout*. Manual de prácticas de la asignatura "Diseño electrónico asistido por ordenador de 3º Ingeniería Técnica Industrial: Electrónica Industrial Universidad Carlos III de Madrid.